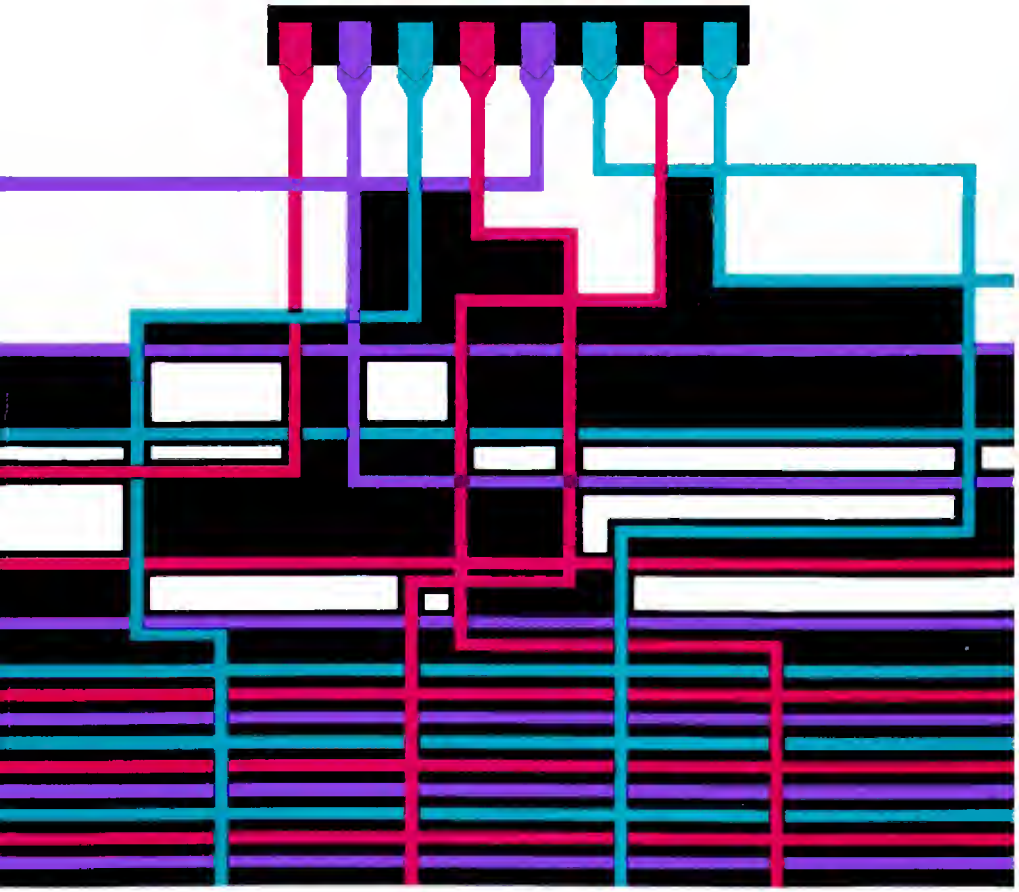
TRS-80® MORE THAN BASIC

BY JOHN PAUL FROEHLICH



BLACKSBURG CONTINUING EDUCATION SERIES™
edited by Larsen, Titus & Titus

The Blacksburg Continuing Education™ Series

The Blacksburg Continuing Education Series™ of books provide a laboratory—or experiment-oriented approach to electronic topics. Present and forthcoming titles in this series include:

- Basic Business Software
- Circuit Design Programs for the TRS-80
- Design of Active Filters, With Experiments
- Design of Op-Amp Circuits, With Experiments
- Design of Phase-Locked Loop Circuits, With Experiments
- Design of Transistor Circuits, With Experiments
- Design of VMOS Circuits, With Experiments
- 8080/8085 Software Design (2 Volumes)
- 8085A Cookbook
- 555 Timer Applications Sourcebook, With Experiments
- Guide to CMOS Basics, Circuits, & Experiments
- How to Program and Interface the 6800
- Microcomputer—Analog Converter Software and Hardware Interfacing
- Microcomputer Interfacing With the 8255 PPI Chip
- NCR Basic Electronics Course, With Experiments
- NCR Data Communications Concepts
- NCR Data Processing Concepts Course
- NCR EDP Concepts Course
- PET Interfacing
- Programming and Interfacing the 6502, With Experiments
- 6502 Software Design
- 6801, 68701, and 6803 Microcomputer Programming and Interfacing
- 6809 Microcomputer Programming & Interfacing, With Experiments
- TEA: An 8080/8085 Co-Resident Editor/Assembler
- TRS-80 Interfacing (2 Volumes)

In most cases, these books provide both text material and experiments, which permit one to demonstrate and explore the concepts that are covered in the book. These books remain among the very few that provide step-by-step instructions concerning how to learn basic electronic concepts, wire actual circuits, test microcomputer interfaces, and program computers based on popular microprocessor chips. We have found that the books are very useful to the electronic novice who desires to join the "electronics revolution," with minimum time and effort.

Additional information about the "Blacksburg Group" is presented inside the rear cover.

Jonathan A. Titus, Christopher A. Titus, and David G. Larsen
"The Blacksburg Group"

TRS-80®

More Than BASIC

by

John Paul Froehlich

Howard W. Sams & Co., Inc.
4300 WEST 62ND ST. INDIANAPOLIS, INDIANA 46268 USA

FIRST EDITION
FIRST PRINTING—1981

International Standard Book Number: 0-672-21813-5
Library of Congress Catalog Card Number: 81-52158

Edited by: *Richard Krajewski*
Illustrated by: *T. R. Emrick*

Printed in the United States of America.

Preface

Only recently has it been possible to manufacture a computer that is truly portable. Thus the microcomputer revolution has been launched, and society must learn to cope with this latest technological rocket. Computer literacy is as necessary for survival as reading, writing, and arithmetic. The literacy is obtained, like other skills, through exposure and practice. Learning to use the computer and learning to program the computer are two different activities. Using the computer with prepackaged programs that provide specific functions is definitely an advantage, but to know how to program the computer to solve each unique problem is the level of skill I am addressing.

Programming may be accomplished by using the various programming languages available for computers. Some of these languages are transportable; that is, they are recognized by different machines with few changes, if any. BASIC, Pascal, FORTRAN, and COBOL are examples of such transportable languages. The skill I want the reader to acquire is more fundamental, for it involves programming in the instruction code of the processor. The instruction code is what interprets the higher level languages, such as BASIC, to a level that the processor can understand. In this case, a microprocessor will be studied because of its small size. In general, the instruction code is not transportable.

What are the advantages of using instruction code? There are many, for instance, speed of execution and conservation of memory. But in my opinion, the most important advantage lies in the fundamental understanding of how a computer operates, which is necessary in the use of instruction code. The concepts of programming and the skills acquired can be applied in all programming situations from the application of the transportable languages to the use of those esoteric codes defining the instruction set of a unique microprocessor. This understanding makes programming in any of the computer languages much easier, and the mystery of what is happening inside those boxes that contain computers becomes clear. The relationship between hardware and software as an integrated system is made understandable. Equipped with that knowledge, you are better able to design dedicated systems to perform specific applications using the new technology for intelligent and flexible control.

To learn this skill, it is necessary to have a development system. A development system is just that—its function is to help you develop systems. Generally, a particular microprocessor must be chosen for the controller to be used in the final design. It is helpful, but not necessary, to have this same microprocessor as part of the development system. If it is the same, a simulator to simulate the desired controller is not necessary. In this case, I have chosen the Z-80 microprocessor manufactured by the Zilog Corporation. It is also available from several other manufacturers. The instruction codes of this microprocessor are also executable to a large extent, but not fully, on the 8085 and 8080 microprocessors manufactured by the Intel Corporation. The popularity of the Z-80 microprocessor makes it an ideal choice for a system controller. The development system is obtained by altering a Radio Shack TRS-80 Model I or Model III. We are fortunate in that the TRS-80 uses the Z-80 as its controlling microprocessor. The alteration to the TRS-80 can be accomplished by loading object code from a tape cassette or disk, or by removing the operating system in the read-only memory supplied with the BASIC machine, and replacing this memory with preprogrammed memory containing the object code that supports the FROLIC development system.

A microprocessor does not make a development system, but a monitor to exercise the controlling microprocessor provides us with a tool to aid in writing software. Such a monitor is described in this book. Chapter I provides a detailed description of the hardware of both the Model I and Model III systems. Chapter 2 describes the FROLIC monitor and the commands available to the user with this monitor. Some hardware is described that allows the inclusion of a single-stepping feature, an attractive additional command to aid in system development. The ability to obtain written text or "hard copy" used to trace system development is provided in Chapter 3. Chapter 4 describes the hardware for programming erasable programmable read-only memory (EPROM) devices. This is a state-of-the-art programmer with features that are not normally found on the commercial equivalents at one hundred times the cost. The source code for driving the programmer for programming the most popular EPROMs is also given. In addition, the appendixes provide all the source code for the monitor. You are encouraged to use this code in any manner you deem appropriate. With a little study and acquired skill, it is hoped that you will alter the monitor to customize it for your own applications.

JOHN PAUL FROEHLICH

Acknowledgments

A special thanks goes to Lewis Winston Grower of United Technology, who helped with the presentation of the material in this book. Others who also aided are Jon and Christopher Titus, of the Blacksburg Group, Katherine Delventhal, Nancy Klock, Scott Mathein, Lenard Lazar, Anthony Lincoln, and Ralph Zegarali. In addition, acknowledgments are extended to those students at the University of Hartford who have used the monitor and made suggestions resulting in the format as described in this book. Finally, gratitude is given to Jack Summers, who provided the core software for the EPROM programmer. I bear the full responsibility for the contents and organization.

JOHN PAUL FROEHLICH

Contents

CHAPTER 1

CHAPTER 2

CHAPTER 3

CHAPTER 4

APPENDIX A

APPENDIX B

APPENDIX C

APPENDIX D

APPENDIX E

APPENDIX F

APPENDIX G

APPENDIX H

CHAPTER 1

More Than BASIC

One wonders about the number of applications for which the TRS-80* system can be utilized. Two models are of particular concern. These are the Model I and Model III, as shown in Figs. 1-1 and 1-2. As originally conceived by Radio Shack, the computer was constructed to execute programs using the BASIC language. BASIC, in the Radio Shack implementation, is an easy-to-use high-level language. (It is called high-level because while it is more easily understood by humans, it is not directly understandable by the computer. It must be translated into low-level instruction code, also known as machine code.) Each program line is scanned by an interpreter, translated, and then processed at program execution time. This method of execution wastes processor cycles and consumes a relatively large amount of memory. (This can be easily demonstrated by running BASIC programs that require a considerable amount of computation and comparing them with a similar machine code program.) Many applications would benefit if they were programmed in the native instruction set of the processor, rather than in the higher level language that needs an interpreter. This would not only decrease the time of execution, but would reduce the memory requirement necessary to execute the same operation. The BASIC interpreter that consumes a considerable portion of memory would no longer be needed. Though machine code is more difficult to write than BASIC, the speed and memory factors make using the instruction set worth the extra effort.

A way to reduce the awkwardness of writing in hard-to-understand machine code (which is made up entirely of numbers) is to

*TRS-80 is a registered trademark of the Tandy Corporation.

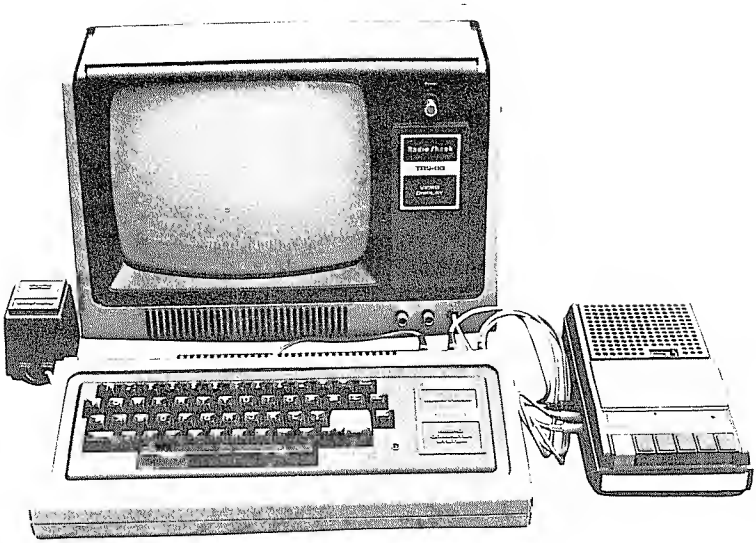


Fig. 1-1. TRS-80 Model I microcomputer system.

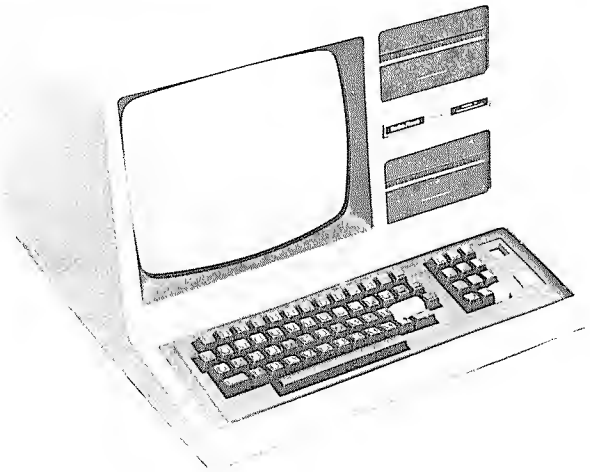


Fig. 1-2. TRS-80 Model III microcomputer system.

use assembly language. Assembly language assigns a very short mnemonic, or abbreviation, to every machine code instruction. This way, you can use the abbreviations instead of the numbers. An interpreter for the assembly language, called an assembler, translates the

abbreviations into machine code. Most of the speed and memory of straight machine code is retained while writing difficulty decreases.

Unfortunately, Radio Shack does not fully support the use of their system for machine code or assembly language programming. At the time of this writing, the programs that are available consist of T-BUG and an editor/assembler (EDT/ASM) for the TRS-80 series which includes the Model I and Model III, Level I and Level II machines.

T-BUG is a rather primitive monitor that allows the execution of machine code. This monitor is extremely sensitive to operator error, and after several hours one tires of its awkwardness.

The extremely versatile editor/assembler available from Radio Shack, which is used to write assembly language programs, is hampered because the execution of this program relies on the tape system. As a consequence, the editing and assembling process is quite slow, particularly for the Level I system because of its slower cassette data rate of 250 bits/second. The Level II system has the advantage of increased cassette speed, which is extended to 500 bits/second. Still, the use of this system for program development is far from desirable. Conditions improve considerably if one is fortunate enough to own an expansion interface that includes the disk operating system. However, the editor/assembler is not available from Radio Shack in a form that will operate with this system, but it is available from sources other than Radio Shack, thus allowing the use of the disk system in program development. It should be noted that Radio Shack does supply an improved monitor with the disk system (D-BUG). However, it has one major flaw: it does not execute programs in real time. This makes it extremely difficult to debug critical time-delay loops. (This is explained later.) Monitors are available from other sources and many are quite good. It was with some of this software that the system presented was developed.

The approach in this book is entirely different. The intent is to provide a complete turnkey development system that rivals those supplied by the major microprocessor manufacturers. (A turnkey system is one that is so complete that all the user needs to do is "turn the key" to begin operation.) To have such a turnkey operation it is necessary to alter the read-only memory in the TRS-80 system. If such a "drastic" modification is not desired, then the user may load the software provided into a convenient set of locations and still have a powerful system. It seems that the only disadvantage of this approach is the additional overhead of loading the operating system every time it will be used. If a user chooses to alter the read-only memory (a procedure described later in this chapter) he should be aware that before any repairs are made on the com-

puter, the original read-only memory (ROM) must be replaced in the keyboard unit before the equipment is returned to a Radio Shack service center. Radio Shack charges an additional fee for servicing if the seal on the keyboard unit has been broken. I have found the need for service on the keyboard unit to be rare. One can always use the software approach or use an external ROM technique. The advantage of placing the monitor in ROM is one of instant availability as soon as the system is turned on; otherwise you must rely on the system-command loading procedures, an adequate compromise. If one is sure he has a reliable unit, the ROMs in the keyboard unit may be replaced with the development system ROMs, which now provide turnkey operation.

The reader is free to determine the extent of modification to be performed on the TRS-80 system. In order to help you to create a development system from your TRS-80, the architecture of various models is described.

As received, a Model I or Model III of the TRS-80 consists of a keyboard with electronics and a video monitor. In addition, the Model I includes a cassette-style tape recorder. The major difference between these two models is that the Model III is a self-contained unit with provision made to add floppy disks internal to the system. The memory structure of both models is exactly the same with respect to memory allocations (memory-mapped video and keyboard, read/write memory, ROM, etc.); however, the units differ in level structure by the amount of ROM actually installed in them. If the unit is a Level I system, the BASIC interpreter ROM occupies the lowest 4K of memory space, and the next 8K of memory is uncommitted. In the Level II unit the BASIC interpreter ROM occupies the first I2K of memory space. The Model III TRS-80 uses, in addition, the next 2K of ROM starting at memory locations 3000H to 3800H (H = hexadecimal) for system utilities. With the exception of the tape storage and retrieval system, input and output operations are accomplished through memory locations. That is, the data from the keyboard is entered into the computer by reading memory locations 3800H through 38FFH. Data generated by the processor is displayed to the video screen by writing into locations 3C00H through 3FFFH. In general, the second half of the fourth 4K block of memory starting with 3800H through 3FFFH is used for input and output. In addition, the Model I requires 32H locations immediately following 3800H for communicating with the disk, real-time clock, cassette relay output latch, and the printer interface. Not all of these locations are used by the system, but the placement of the memory references effectively removes this space from being available for general-purpose use.

Memory space starting at 4000H and above is available for general memory purposes. The 4K machines have 4K dynamic read/write (R/W) or random-access memory (RAM) devices occupying 4000H to 4FFFH, and 16K machines have 16K dynamic read/write devices occupying 4000H to 7FFFH. In both models the total memory expansion capability is to 48K. The Model I microcomputer can readily be expanded from 4K to 16K by a slight internal alteration to the keyboard unit which involves changing the 4K R/W dynamic memories (a set of 8 IC chips) to 16K devices, and the reconfiguration of 2 DIP shunts. The DIP shunts can be replaced with dual in-line switches to allow for making these changes. Further expansion to the memory is possible with an expansion interface that can be your own design or the expansion unit available from Radio Shack. The expansion unit attaches to the TRS-80 using the 40-line edge-card connector. Memory expansion for the Model III unit is much simpler, since the memory expansion capability is part of the desk-top unit.

To help clarify the memory structure of the two models, a memory map of the TRS-80 system is shown in Table 1-1.

As stated earlier, the significant difference between the Level I and Level II systems is the 12K BASIC read-only memory (ROM). The 12K ROM supports an expansion interface and increased tape speed for program storage from 250 b/sec (bits-per-second) for the Level I machine to 500 b/sec for the Level II machine. In addition, the Model III can operate at tape speeds of 500 b/sec and 1500 b/sec. If BASIC is "your language," then the ROM as supplied by Radio Shack should be considered because the disk BASIC does not use all of the utilities or subroutines provided in the ROM. A total commitment to the Radio Shack system must be considered if extended BASIC programming is required.

There is no intention to use BASIC in the development system that will be described here, so it will not be discussed further. Attention will be focused on the architecture of the system that is the same whether it is a Model I or a Model III system.

Some of the differences and similarities between the various TRS-80 models have been described. More details of these models will be given later in this chapter. The important similarity that we wish to emphasize is that the computational "heart" of the two systems is identical. The computational device employed is a device called a microprocessor. This device is a complex design of thousands of transistors and resistors in an economical, and physically small, package. The TRS-80 exploits the microprocessor's computational characteristics to the extent that the TRS-80 is, indeed, a microcomputer.

Industrial pressures did not cause the development of the micro-

Table 1-1. Memory Maps of Model I and Model III TRS-80 Systems

Hexadecimal Address	Model I		Model III	
	Level I	Level II	Level I	Level III
0000-0FFF 1000-1FFF	4K BASIC ROM	12K BASIC ROM	4K BASIC ROM	12K BASIC ROM
2000-2FFF 3000-3FFF 3000-37DF 37E0-37FF		↓		↓ 2K ROM for System Utilities
	Memory-Mapped 37E8 Printer	Memory-Mapped 37E0 Disk Select 37E4 Relay 37E8 Printer 37EC CMD/Status 37ED Track 37EE Sector 37EF Data	Memory-Mapped 37E8 Printer	Memory-Mapped 37E8 Printer
3800-38FF	Keyboard Matrix	Keyboard Matrix	Keyboard Matrix	Keyboard Matrix
3C00-3FFF	Video Display	Video Display	Video Display	Video Display
	Start of Read/Write Memory			
4000-4FFF	4K Dynamic RAM Ends at 4FFFH			
5000-5FFF	8K Dynamic RAM Ends at 5FFFH			
6000-7FFF	16K Dynamic RAM Ends at 7FFFH End of Available RAM in Model I			
8000-8FFF	32K Dynamic RAM Ends at BFFFH Model I Requires Expansion Interface			
C000-FFFF	42K Dynamic RAM Ends at FFFFH			

processor to be used for the development of home computers. Industrial pressures were focused at producing a device that would simplify the development and/or increase the intelligence of controllers while simultaneously reducing costs. The purpose of an intelligent controller is to reduce the complexity and cost of performing a specific operation. In industry, the less complexity involved, the greater the reliability and the greater the ease of manufacture.

To keep the cost of an intelligent controller low, most controllers do not include a full ASCII keyboard, a crt monitor, nor the BASIC language. Instead, a limited amount of external controls and indicators, if any, are usually involved. Instead of BASIC programming, only the machine language of the microprocessor is used. The use of the machine language of the microprocessor is more primitive in its structure when compared to BASIC. However, it is more flexible, faster, can be made to operate in real-time, and uses far less

memory for a programmed function. (Remember that the Level I uses 4K of memory and the Level II uses 12K of memory alone to hold the BASIC interpreter and the system utilities.)

Some examples of intelligent controllers using microprocessors are found in heating controllers, automated test equipment, automotive products, sewing machines, microwave ovens, and the tremendous variety of hand-held electronic games presently appearing on the consumer market. When industry decides to build any of these devices, a development system is usually employed. A development system allows the designer to develop and test the software used to control a microprocessor, to emulate the microprocessor, and to examine the execution of the control function in process. Typically a development system runs into thousands of dollars. There is, however, an inexpensive solution.

The solution is to modify a TRS-80 for the following reasons. First and foremost is that the microprocessor used in the TRS-80 system is a Z-80, designed and manufactured by the Zilog Corporation and second-sourced by the Mostek Corporation. Many of the Z-80 microprocessor instructions are directly compatible with the Intel 8080 and 8085 microprocessors, second-sourced by many companies. These three devices are used extensively by industry in control applications. It is true that both the 8080 and 8085 have fewer instructions, and there are hardware differences between the Z-80 and the Intel devices. However, there are more similarities than differences, and the ease of making adjustments encourages the use of the development system to be described, whichever processor is chosen. The software and optional hardware features which are presented in this book allow the system to have all the capabilities of a comprehensive development system. For turnkey operation this involves the purchase of erasable programmable read-only memory (EPROM) at an additional cost of about $20. (A preprogrammed version of this EPROM is available. The address of the supplier is given in Appendix C.) The last consideration is the availability of the data, address, and control lines at the edge-card connector of the keyboard unit. It is also possible to implement the TRS-80 unit in the actual control application and utilize its internal Z-80 as the controlling microprocessor. If this is done, the crt and keyboard are available for the small investment in the Radio Shack TRS-80 computer which includes 4K RAM and either 4K, 8K, 12K, or 14K ROM capabilities.

To take advantage of the TRS-80 as a development system, one should be familiar with the Z-80 processor. The best source for this information is one of the manufacturer's technical reference manuals for this device. (See Appendix B.) The link of the Z-80 micro-

processor to the external world with the TRS-80 is either the 40-line (Model I) or the 50-line (Model III) edge-card connector. The signals present on these edge-card connectors are listed in Tables 1-2 and 1-3. The use of these connectors is one of the most significant differences between the Model I and Model III TRS-80 Radio Shack systems. Historically, the Model I system was first available. Therefore, the Model I system is first covered in the "in-depth" discussion which follows. The edge-card connector is presented first, and then the details of the port assignments, video access, graphic capabilities, keyboard input, and information storage and retrieval are presented. The Model III description follows; it concentrates on the significant differences between the two models.

MODEL I TRS-80

Shown in Table 1-2 are the signals present at the 40-line edge-card connector. Most of the lines are buffered, and it may be desirable to buffer those remaining lines if you are going to interface to this connector. Let's examine each of the terminations at this edge connection.

The terminations from the processor board fall into the following categories:

1. Data
2. Address
3. System Control
4. Interrupt

Data

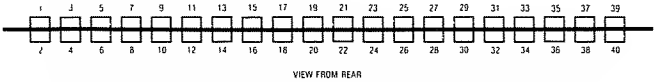
The first terminations to be considered are the data lines. These eight lines are bidirectional, and therefore data is available from the processor or made available to it. These lines are buffered using 74LS367 (DM8097) TTL circuits, which are hex three-state noninverting buffers. As the buffers are three-state, the TEST line controls the three-state function while the microprocessor determines the direction of the data flow. Even though the TEST line has control over the data buffers, this is not its only function.

Address

There are 16 address lines which are all output lines. This gives the processor the capability of addressing 65,536 locations. The same type of buffer is used on these lines, and the activation is controlled by the TEST signal.

Table 1-2. Edge-Card Connections for Interface Signals Available From the TRS-80 Model I System

Pin	Signal Name	Description
1	RAS/	Row Address Strobe Output for 16-Pin Dynamic RAMs
2	SYSRES/	System Reset Output, Low During Power Up Initialize or Reset Depressed
3	CAS/	Column Address Strobe Output for 16-Pin Dynamic RAMs
4	A10	Address Output
5	A12	Address Output
6	A13	Address Output
7	A15	Address Output
B	GND	Signal Ground
9	A11	Address Output
10	A14	Address Output
11	AB	Address Output
12	OUT/	Peripheral Write Strobe Output
13	WR/	Memory Write Strobe Output
14	INTAK/	Interrupt Acknowledge Output
15	RD/	Memory Read Strobe Output
16	MUX	Multiplexer Control Output for 16-Pin Dynamic RAMs
17	A9	Address Output
1B	D4	Bidirectional Data Bus
19	IN/	Peripheral Read Strobe Output
20	D7	Bidirectional Data Bus
21	INT/	Interrupt Input (Maskable)
22	D1	Bidirectional Data Bus
23	TEST/	A Logic 0 on TEST/ Input Three-States A0-A15, D0-D7, WR/, RD/, IN/, OUT/, RAS/, CAS/, MUX/
24	D6	Bidirectional Data Bus
25	A0	Address Output
26	D3	Bidirectional Data Bus
27	A1	Address Output
2B	D5	Bidirectional Data Bus
29	GND	Signal Ground
30	D0	Bidirectional Data Bus
31	A4	Address Bus
32	D2	Bidirectional Data Bus
33	WAIT/	Processor Wait Input, to Allow for Slow Memory
34	A3	Address Output
35	A5	Address Output
36	A7	Address Output
37	GND	Signal Ground
3B	A6	Address Output
39	GND	Signal Ground
40	A2	Address Output



NOTE: / means Negative (Logical 0) True Input or Output

System Control

The next four signals control the direction of data flow to/from the processor at the correct time in the processor cycles. They are obtained from the CPU control system signals originating from the Z-80 microprocessor. These signals, as defined by the Zilog Corporation, are memory request (MREQ/), input/output request (IORQ/), read (RD/), and write (WR/). These are not the signals at the 40-pin edge connector, but are signals generated by the Z-80 chip. (*The '/' is used to indicate active-low, logic zero assertion of these signals.*) Proper gating of these lines gives us the four TRS-80 bus control signals: read (RD/), write (WR/), input (IN/), and output (OUT/). These signals are active, or asserted, in the low state, thus the "slash" (a convention used throughout this book to indicate an active low condition). The RD/ and WR/ are used for memory reference instructions; the IN/ and OUT/ are used for peripheral control to port locations. These signals make interfacing to the computer bus a minimal effort.

Interrupt

The next lines considered are the interrupt (INT/) and interrupt acknowledge (INTAK/). As delivered, the TRS-80 system allows using only one of the three interrupt modes that the Z-80 microprocessor supports. The expansion interface that supports the disk operating system contains a real-time clock that provides a 25-ms interrupt to the INT/ line for some real-time applications. The TRS-80 system uses this signal to support a 24-hour clock. It is important to realize that if the expansion interface is connected and the interrupt is enabled through a software command, a system "crash" will result if software to support the requested interrupt is not present in memory. Radio Shack has provided the software on the Level II system to support the clock. However, once you modify the system with the new software (a software modification here, not hardware) a potential loss of control is possible. The single-step feature in the FROLIC monitor uses the interrupt to end execution of a single instruction. The expansion interface 25-ms clock must be altered to prohibit interrupts if the single-step option is desired.

Having the full interrupt features of the Z-80 microprocessor is not necessary for successful use of the FROLIC monitor. However, in some applications, the availability of the full interrupt features may be useful.

Two interrupt modes that the Z-80 uses cause a vector to the address 0038H. These are the mode 0 and mode 1 interrupts. At power-up and reset, mode 0 is automatically enabled. Even though the

mode is established, the microprocessor will not respond to interrupts unless the interrupt enable instruction (EI) is executed using the proper software command. With mode 0, it is necessary to strobe an FFH onto the TRS-80 data bus using the INTAK/ to cause a vector to 0038H. Up to eight restart vectors may be strobed on the data lines. In fact, the instruction does not need to be one of the restart instructions (RST), but any valid instruction in the Z-80 instruction set may be used. Further details can be obtained from the Z-80 technical manual. However, by changing the mode of interrupt from 0 to 1 using the appropriate instruction (IM 1), it is not necessary to strobe an instruction on the data lines. The processor automatically vectors to address 0038H. The TRS-80 only supports the Z-80 mode 1 interrupt, and the Z-80 only vectors to the RST 38H address.

The strobing of instructions or address data into the processor from a peripheral is one of the functions of the INTAK/ line. Its function is identical to that of the RD/, except that the source of the data is not from a memory reference. If mode 0 is used, you should not expect an FFH to be strobed on the data lines by default without supplying the necessary hardware using the INTAK/ to guarantee its presence.

The third mode of interrupt (mode 2) is a vector type, but in this case the lower 8 bits of the address of a pointer where the actual interrupt program resides are strobed on the data lines using the INTAK/ signal. The high-order address of the pointer is obtained from data present in a special register (I register) in the microprocessor. Instructions to load the high-order vector in the I register and to indicate a mode 2 interrupt must be executed before any interrupt using this mode is executed.

Another function of the INTAK/ line is to indicate to the peripherals that an interrupt has occurred. This line was provided in the TRS-80 design so that interrupt flags can be cleared without software overhead. As we see, it can also be used if the TRS-80 is modified for mode 0 or mode 2 interrupts to perform the strobing of data as just described.

To take advantage of the first and third modes of interrupt on a Model I TRS-80 system, a slight modification must be implemented within the keyboard unit. It involves the alteration of the TEST/ line input circuitry. Later in this chapter a list of the modifications that can be made to the Radio Shack Model I and Model III systems will be given. To use these modes after the hardware modifications have been made, it is necessary to provide the software support that will alter the mode and load the interrupt vector if required.

Additional Control Signals

The WAIT/ line is used to "inject" wait states into the program flow in order to accommodate slow devices. This will not be used, nor considered further. Caution: Holding the wait state line low for a period longer than 0.5 ms will disrupt the normal operation of the dynamic random-access memories.

The SYSRES/ line is used to reset the system. Because of the use of dynamic memory, it was necessary to use the nonmaskable interrupt input for system reset. This will point the microprocessor to a known address when the reset push button to the left of the edge-card connector is depressed. This reset is useful when a system "crash" occurs. The use of nonmaskable interrupt instead of the RESET pin of the microprocessor allows for the refreshing of dynamic memories without data losses.

The TEST/ line isolates the Z-80 from the external devices. It places all data and address lines, the RD/, WR/, IN/, OUT/, RAS/, CAS/, and MUX/ signals in the high impedance "third" state. The TEST/ line may be modified in the development system, but it is easily restored if servicing is required.

The row-address strobe (RAS/), column-address strobe (CAS/), and multiplex control line (MUX/) are used to expand memory using dynamic random-access memories (RAM). It is quite easy to interface the system to dynamic RAM because of the presence of these signals.

Absent Control Signals

One of the signals absent on the edge connector is halt acknowledge (HALT/). It is impossible to use the assembly-language HALT instruction in an unmodified TRS-80 system, for the designer of the Radio Shack system has taken the HALT signal and ORed it with the RESET signal, and the output of this gate goes to the nonmaskable interrupt pin of the microprocessor. This results in resetting the system if a HALT instruction is executed by the microprocessor.

Another signal absent from the edge-card connector is the line which indicates that the machine is in a fetch state of the machine cycle (MI/). This line is useful for single-step hardware modification, but not essential. A reference to this line occurs at the end of Chapter 2. If the use of this line is desired, an additional modification is required and is given in the section dealing with the single-step implementation.

A ground is provided in several locations. In older units, +5 V dc is accessible at pin 39; however, a modification by Radio Shack re-

moves this power connection in newer models. In any event, you should not attempt to use the TRS-80 +5 V for any external circuits. An external power supply should be used for such external circuits, along with a common ground connection to the TRS-80.

This completes the discussion of the signals available at the edge-card connector. It is helpful to discuss features that involve hardware peculiar to the TRS-80. These include port assignment and functions, video, graphic capabilities, keyboard, information storage and retrieval by tape, and lower-case conversion possibilities. Some schematics are presented, but these illustrations are not detailed. In many cases, the schematics are modified in order to aid understanding. Functional purpose is the primary concern. This is necessary if one wishes to understand the instructions contained within the monitor as listed in the Appendixes. If this is of no concern, skim this section and go to the modification description.

Port Assignments

Only one port is assigned in the Model I TRS-80 system, and that port is FFH, which is used for both input and output. As an output port, the TRS-80 uses bits 0 and 1 of the port bus to produce a square wave of 0.85 volt peak amplitude. This signal is applied to the input of a cassette recorder by way of the 5-pin DIN connector. Actually, any audio recorder may function as a recording device. The logic on bits 0 and 1 of port FFH determines the voltage level. A 0.46-volt level is established by outputting zeros on these two lower-order bits (bit 0 = 0 and bit 1 = 0). When the two bits equal 01 (base 2), a 0.85-volt level is produced. When the two bits equal 10 (base 2), a 0.00-volt level is produced. The 11 (base 2) condition is not used. Bit 2 of port FFH is used to control the motor of the recorder, and when high it activates a reed relay switch, turning on the recorder. Bit 3 is the last bit of this port to be used by the TRS-80 system. This bit allows control of the large graphic or alphanumeric presentation to the video screen. Normally 1024 locations are used for the display of data written to the screen R/W memory located from 3C00H to 3FFFH. In the enhanced display, only the even memory locations are used, and the data characters residing in these locations are extended horizontally and occupy two positions on the video screen. Therefore, only 512 locations are visible, and 16 lines of display data are possible.

As an input, port FFH uses only two bits. Bit 7 is used for reading the cassette. Bit 6 contains the information for enhanced display mode. If enhanced horizontal presentation is currently activated, then this bit is 1. Otherwise, it is 0, indicating normal screen presentation.

Video Output

Information can be "viewed" by the user by writing to memory locations 3C00H to 3FFFH. The data located at these addresses is shared by the crt and the microprocessor. Through the use of special decoding ROMs, this data is converted to familiar characters and is displayed on the monitor screen. The characters shown are those defined by the American Standard Code for Information Interchange (ASCII). Without modifications, only upper-case characters, numerals, and punctuation marks are available. In addition there is a set of graphic "characters." The pattern of a graphic "character" is determined from the byte written to the shared video R/W memory. If the most significant bit (7) is active or high, the resulting display is graphic; otherwise, the display is alphanumeric. The six lower bits of the byte written to the screen R/W memory located from 3C00H to 3FFFH determine the pattern on the video screen. This memory space results in 16 lines, each having 64 character or graphic positions. If bit 7 is a logic 1, the remaining six bits can generate one of 64 possible graphic characters within each of the 1024 graphic block positions. As the TR-80 Model I is delivered, there are only 7-bit words available for storage in the video RAM. The "missing" bit (6) is not used. Therefore, the patterns and characters formed on the display are "repeated" and thus are not dependent on the absent bit. Fig. 1-3 shows how the six bits determine the screen pattern. With the aid of the monitor, you can write graphic words to the video addresses. This is a good method of seeing the graphic capabilities of the TRS-80.

Radio Shack provides a lower-case modification for the Model I that adds the "missing" memory bit and expands the video word to a full byte. Also included in this modification is a replacement

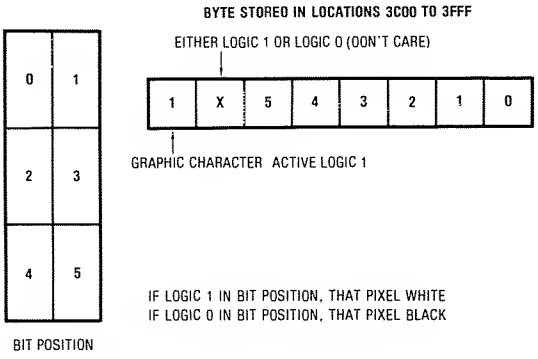


Fig. 1-3. Graphic patterns formed with the six least significant bits.

for the video character-generating ROM, which provides for downward extension, thus allowing characters such as y and g to extend below the line. You can add your own modification by providing the additional memory and a switch to activate it, thereby obtaining the full display features available in the ROM. However, one must replace the ROM character generator with the updated version to obtain the automatic extension feature. It is suggested that if lowercase is required, you purchase the LC modification kit from Radio Shack. The feature may be nice but it is not essential to the development system.

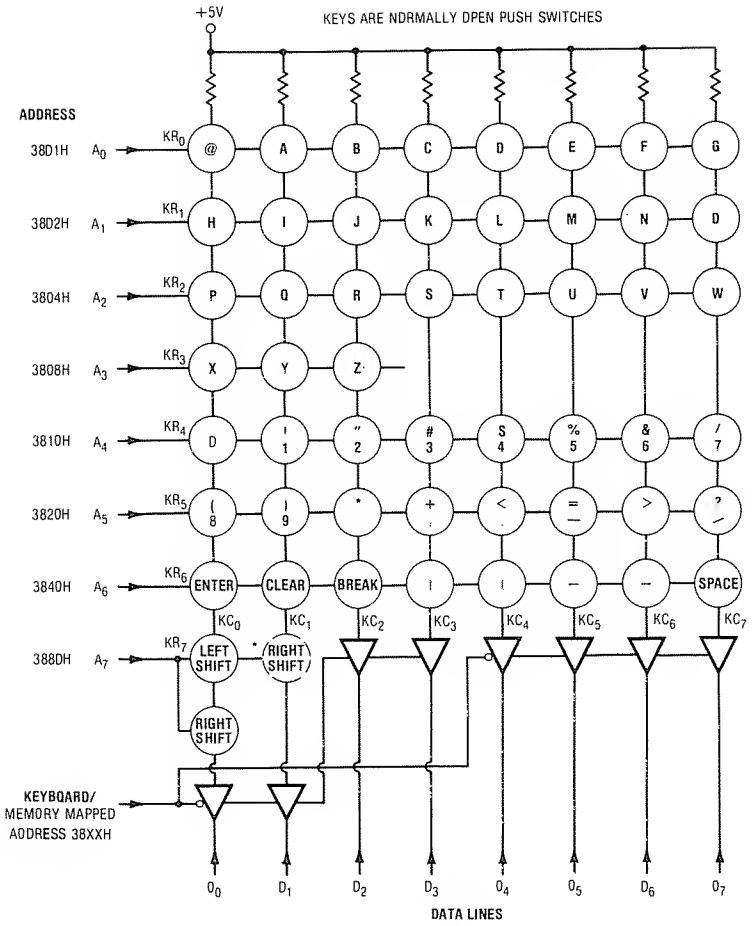
Keyboard Input

The keyboard is a matrix device, and the layout of the keys in relationship to the data and address lines is shown in Fig. 1-4. Some logic has been omitted from the schematic so as not to clutter the presentation. The most important feature of this arrangement is the many possibilities for decoding. In fact, one page (256 bytes) of memory is lost to this area, but this results in many unique codes (upwards of 200). The addressable memory space that is used to detect a response on the data lines starts with address 3801H and continues to 38FFH. To obtain a unique code, the programmer must be careful. For example, the addressing of memory 38FFH results in the possibility of any key producing a response, since all eight address lines will be active, whereas addressing 3808H will only produce a response if the X, Y, or Z key is depressed. The matrix of the keyboard is such that a unique line is accessed by the following address excitations: 3801H, 3802H, 3804H, 3808H, 3810H, 3820H, 3840H, and 3880H. Since location 3880H has only one key attached, the SHIFT key, accessing this location may be used for quick unique exits from a software loop. An example of the short assembly-language code required to detect this key closure is shown:

```
EXIT  LD A, (3880H)    ;READ DATA AT LOCATION 3880H
      OR A             ;SET ZERO FLAG AND
      RET Z            ;RETURN TO LOOP IF NONZERO
                       ;START OF ESCAPE
```

If one wishes to see if any key is pressed, replacing 3880H with 38FFH accomplishes this.

It is obvious from the schematic which keys will produce data on the data lines (when pressed) if their particular line is accessed. It is the function of the key-scan routine to eliminate "bounce." (Mechanical switches take time to settle, and the processor speed in executing code is such that contact bounce, or repeated opening and closing, results in producing multiple readings of the key unless corrective action is taken.) Another function of the scanning



*Model I: Right shift, left shift on data line D.
Model III: Left shift data line 0, right shift data line 1.

Fig. 1-4. Keyboard matrix for Model I and Model III microcomputer systems.

routine is to produce a code to represent the key. The American Standard Code for Information Interchange (ASCII) format is the usually accepted form.

Tape Input/Output

A few words on the tape format as established by Radio Shack for the Level II systems follow. The rationale for this format is

sound, but problems in saving data do arise due to head alignment and improper recording and playback levels. Careful attention must be paid to these levels.

Information saving and retrieval in the TRS-80 system uses magnetic recording media with a standard audio cassette as the storage element. Certified computer tape improves the reliability, but exchange of information between systems sometimes presents a problem. It is also useful to bulk-erase a cassette before it is reused to store new information.

Data is recorded at the rate of 500 b/sec. This rate corresponds to approximately 62 bytes per second. To maintain compatibility with existing object code tapes produced by Radio Shack and other sources, this format is used for cassette storage in the FROLIC monitor program.

Data is stored on the tape by a series of pulses. The pulses are spaced every 2 ms, which establishes the frame for written data. As you probably know, the data is in serial form, one bit after the other, and consists of representations for logic 1 and logic 0. To record a logic 1, an additional pulse is recorded within the 2-ms time period. A center position of 1 ms is chosen. The absence of a pulse indicates a logic 0, and, consequently, both logic conditions may now be represented. The pulses are generated for recording using the two least significant bits of a data byte written to port FFH. The reference level is established from the output of 00 in these bit positions. This produces a dc level of 0.46 volt. A pulse is formed by the output of three consecutive bit patterns that change the reference level first to 0.85 volt, then to 0 0 volt, with a final return to the 0.46-volt reference. The upper and lower voltages are each held for approximately 0.1 ms. Therefore, the pulse is formed by output of a 01, a short delay, output of a 10, another short delay, and finally an output of 00. (In this illustration, only bits 0 and 1 of the port output are given.) All of these data patterns are written to port FFH with bit 2 active to maintain the recorder in a running state. Fig. 1-5 shows the levels associated with the output patterns in relation to the pulses and pulse separation for the transmission of both a logic 0 and a logic 1.

Tape Data Format

In order that this technique of recording data to cassette tape be successful, it is necessary to synchronize the data and establish a format so that the receiving device can process the data in an acceptable manner. In addition, an error-detection byte is added to provide simple indication of a valid transfer from tape to CPU. The following scheme is used for writing data on the cassette re-

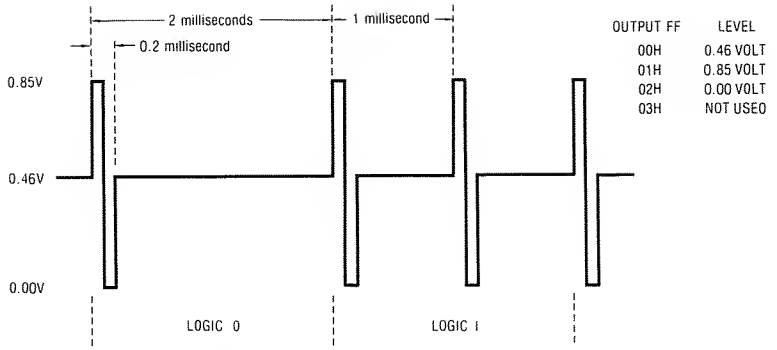


Fig. 1-5. Data pulses for cassette output.

corder. First, the recorder is turned on with the 0.46-volt level established. An output of 40H to port FFH starts the write routine. A time delay is necessary to allow the recorder to reach constant speed. One second is sufficient, but generally a longer delay exists on tapes. Logic 1s and 0s are written on the tape, but in order to establish data presence, a synchronization byte is required. The indication that data is about to be available is flagged by the decoding of an A5H, which is the tape synchronization mark. From this time on, synchronization must be maintained, or an unsuccessful write will be the result. Immediately following the A5H synchronization mark is the label mark 55H. The next six bytes of data are used for tape identification. Data must be present to fill these six locations, or synchronization is lost during the read command. After the identification, the next byte must be either a 3CH or a 78H. The 3CH indicates a data block follows. A 78H indicates the execute address of the loaded program to the reading routine.

The data blocks are formatted in the following manner. The first byte following the 3CH indicates the number of data bytes in the block. A 00H means 256 data bytes. This is the maximum for a block length. The minimum block length is indicated by a 01H, which states that only one data byte is present in the block. The next two bytes written to the tape are the low and high addresses which establish the starting location of that data block. The reading routine uses this data to indicate the starting address in memory for deposit. Error detection is provided using a checksum. The checksum is formed by the addition of each byte written to the tape starting with the value of the address. Only one byte is used for the computed checksum. Therefore, all carries are ignored. This sum is stored in memory and updated as the writing of the block continues. Upon completion of a block, the checksum is output as the last data

in the block. It is not included in the checksum computation, but it is compared against the computed checksum. If the two checksums are equal, then data transfer occurred without error (it is assumed). If more data is to be written, the block mark 3CH is used to signal that more data follows.

The last information written on the tape is used to indicate the execute address of the program just written. A 78H is the signal to the reading program that the next two bytes on the tape form the address of execution. Like most addresses, the low byte precedes the high byte. No checksum is used in this block. The last three bytes complete transcription of the tape.

The method used for reading a tape produced in the TRS-80 format just described will require a conditioning circuit to convert the pulses on the tape to logic levels suitable for decoding. The pulse described has a fundamental period of 0.2 ms. As a result, there is a strong fundamental at 5000 Hz. This frequency is suited for the audio cassette with the higher harmonics filtered by the electronics in the recorder. The presence of a pulse on the cassette tape causes a data latch consisting of a set-reset logic flip-flop to be set. If one viewed the output of the cassette on an oscilloscope it would reveal a few cycles forming a tone burst with a fundamental of 5000 Hz. The tone bursts are amplified and rectified. As a result of this process, a negative asserted pulse latches the flip-flop. Any new pulses processed by the electronics will have no effect on the latch unless this latch is cleared. The clearing of the data latch is accomplished using a hardware-software technique. The technique generates the necessary signals that clear the latch. If you study the monitor listing in Appendixes D and E under the section dealing with the cassette-read section, you will notice the instruction OUT (0FFH),A. Any data may be output at this time. However, for the Model I system, the cassette unit must remain on. This requires the presence of a logic I in position bit 2 of the port FFH latch. Since the electronics used for recording are not active during the reading process, the states of the data in bits 0 and 1 are of no consequence. The graphic mode used to control the width of characters written to the crt RAM is controlled by bit 3. This also does not affect the tape function. Therefore, the only bit of consequence in clearing the latch is bit 2, which must remain active

If upon clearing the latch with an OUT (0FFH),A, a read using IN A,(0FFH) results in the presence of a logic I in bit 7, then a pulse passed the read tape head. If logic 0 is in bit 7, then no pulse is present.

The software required to read a data track first looks for pulses or single bits. A "leader" is always written that consists of up to

256 pulses spaced 2 ms apart. The minimum number of pulses required is that which ensures that the cassette has reached its operational speed. To read a TRS-80 tape you must first obtain synchronization. This is accomplished by reading data pulses and storing these pulses by shifting them sequentially into a register. Testing this register for the sync mark A5H after each shift will at some time produce the required synchronization. Once this is established, the data then can be read a byte at a time. The sequence used in the software to read the tape will first search for the identification mark 55H. This mark must be the byte following the sync mark. Following the identification mark are 6 bytes used for a tape label. The data mark 3CH is the next byte in the sequence. It is followed with the byte used to indicate the number of data bytes in the current block. The low address followed by the high address informs the software of the address of the first location at which data is to be stored in memory.

The data bytes are next in sequence. The last byte in the block is the checksum. The calculation of checksum byte started with the loading address and ended with the last data value in the block. The tape reading finishes when the software detects the execute address mark 78H in place of 3CH. The execute address follows in two bytes, but there is no checksum computed for this address.

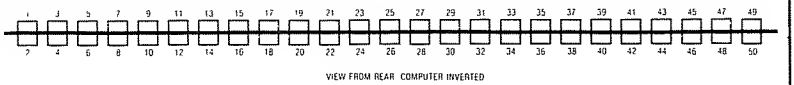
TRS-80 MODEL III

Only the changes in the hardware of the Model III which directly affect the operation of the FROLIC monitor are given in this section. The two models are configured with identical memory mapping, and this contributes to the compatibility between the systems. The differences occur in five areas. The first section covers the Model III bus; the second, the port assignments; the third, the video display; the fourth, the keyboard; and the fifth, information storage and retrieval.

The interface of the TRS-80 Model III to the external world consists of a 50-pin edge-card connector. There is no relationship between this connector and the 40-line edge-card connector used for the Model I. Only those signals for port select and control or reset are included on this edge-card connector. These include address lines, data lines, and control lines. To allow data flow at the edge-card connector, the edge-card connector is made active by an output of 10H to port ECH. This is one of the new ports that is available on the Model III system. Table 1-3 shows the signals available at this edge-card connection. Every signal has a ground associated with the line. Disabling the 50-line connector so that no signals are

Table 1-3. Interface Signals Available From the TRS-80 Model III

Pin Number	Function for TRS-80 or Z-80
1	Data 0
3	Data 1
5	Data 2
7	Data 3
9	Data 4
11	Data 5
13	Data 6
15	Data 7
17	Address 0
19	Address 1
21	Address 2
23	Address 3
25	Address 4
27	Address 5
29	Address 6
31	Address 7
33	IN/
35	OUT/
37	RESET/
39	INT/
41	I/O WAIT/
43	DATA BUS IN/
45	(Not Used)
47	M1/
49	IOREQ/
2-50 (Even)	Ground



VIEW FROM REAR COMPUTER INVERTED

present on the bus may be accomplished by an output of a 0 in the fourth bit of port E0H.

Port Assignments

The following ports are used in the Model III: E0H, E4H, E8H, E9H, EAH, EBH, ECH, F0H, F1H, F2H, F3H, F4H, and FFH. (Ports E0H and ECH are not uniquely decoded. That is, E1H, E2H, and E3H may be used to address port E0H; and EDH, EEH, and EFH may be used to address port ECH.) Not all the functions of these ports are known at this writing, but those relevant to the FROLIC monitor are described. Ports FFH, E0H, and ECH are important to the FROLIC monitor.

Port FFH functions as both input and output. As an output port,

the least significant bits (0 and 1) are used for level control in the recording process as they were in the Model I system. The function of the other bits of this port is not known. As an input port, it appears the function of the least significant bit (0) is pulse detection and timing from the cassette recorder input conditioner. Bit 1 is used as a flag to signal that the cassette motor is operating. Bit 2 is used to indicate that the extended graphic mode is active. (The Model I system used bit 6 as this flag.) Bit 3 shows that the alternate character set is active. Bit 4 is used to show that the external i/o data and control lines are active. Bit 5 is unknown. Bit 6 is always active, but its function is also unknown. Bit 7 is from the data latch. It is set by data pulses from the output of the cassette. It is reset by an input to port FFH, and its function is identical to that in the Model I system. However, the function of bit 0 of Port FFH is different than that of bit 7, in that it is active during data transitions, and not latched as is bit 7. The data latch, you recall, stores flux transitions that pass the read head of the cassette player. These transitions are either the data or separation pulses. However, bit 0 is active only when a pulse is detected. There is retention in a latch unless another transition is detected, at which time the flag is complemented. After a long delay with no flux transitions, the flag is automatically cleared.

Port ECH (EDH, EEH, and EFH) controls some of the peripheral functions of the TRS-80 Model III. The operating state of these peripherals is made known by the input status of port FFH. To activate these functions, outputs to port ECH are made. The cassette recorder motor, the mode that controls the horizontal width of the crt display, the selection of character sets, and 50-line bus activation are all controlled by this port. To turn on the cassette, a 02H is written to port ECH. Likewise, an output of 04H alters the horizonal size, a 08H selects the alternate character set, and a 10H enables the control-data lines for external i/o. An output of 20H alters the status flag of port FFH, but the function of this flag is unknown at this time. The activation of the remaining three bits, 0, 6, and 7, produces no visible effect.

Port E0H is used as a priority interrupt control and status port. The highest priority device is connected to bit 0, and the lowest is connected to bit 7. A logic 0 (active low) indicates a device has requested an interrupt. The status of those devices requesting to interrupt is determined by an input of port E0H. To allow a device to interrupt, port E0H must enable a gate to allow the interrupt to reach the status latch. In addition, the instruction EI (enable interrupt) must have been executed. For an example, an OUT E0H 01H or 02H allows the data pulse from the cassette tape to

produce an interrupt. Depending upon which data was output, the pulse produces the highest or second highest priority. The Model III uses this for tape read operations. An OUT E0H 04H sets bit 3. This allows the real-time clock pulse to interrupt the processor. OUT E0H 08H allows the interrupt from the external bus to reach the processor. The source of other interrupts is not known, but locations in R/W memory make it possible to direct control to any place in memory. The locations for transfer in the unmodified TRS-80 Level II are at 403DH, 4206H, 4209H, 4040H, and 4043H. The addresses are listed in interrupt priority and are levels 3, 4, 5, 6, and 7. In a system of your design, your software should interrogate port E0H, and control transfer should be specified by your program.

Ports E8H, E9H, EAH, and EBH are used for the RS-232C communication link. These function in identical manner to the functions of the RS-232C used in the Model I. An interface is required to take advantage of these ports.

Ports F0, F1, F2, F3H, F4H, and E4H are used for the floppy disk controller. A Western Digital FD1791 is used in this application. Port F4H controls the drive select with the least significant bits used for drive selection. It appears bit 7 of this port controls the density mode, with a logic 1 indicating double density. Bit 6 controls the gating of the data request line (DQR) to the wait pin of the Z-80 processor. This allows the processor to go into a wait state until the byte of data from the disk controller is available. This does not disturb the memory refresh cycle. When the processor examines the software control for disk boot-up, an output is made to port F4H of value C1H. Port F0H is used for command words and status. Port F1H is used for track data, and F2H is used for sector data. Port F3H is used for the data. Port E4 is also used in disk operations. The most significant bit is used. It is assumed that this allows the interrupt of the controller to interrupt the Z-80 microprocessor when it has finished the current command from the disk controller.

Video Output

The changes to the video section are also significant. Both upper-case and lower-case characters are available. (The FROLIC monitor uses only the upper-case values and masks out the new characters. However, by changing the mask, the user has access to the lower-case letters.) Through the use of a port control, a second set of characters is available. The port output functions as a hardware mask, and activation of the alternate character generator is made through port control as will be described in the next section. The characters that are available consist of the standard upper- and

lower-case alphabet, some Japanese symbols or character sets, the Greek alphabet, Spanish, French, and German accents and markings, and other special graphics and symbols. Figs. 1-6 and 1-7 display the complete character sets available. The set shown in Fig. 1-6 occurs when bit 3 of port ECH is low, while the set shown in Fig. 1-7 occurs when bit 3 of port ECH is high. The two sets of characters are displayed, including graphics. Notice there is considerable duplication, particularly in the alphabetic character set. The graphic characters are the same as those used in the Model I, but logic 1 in the

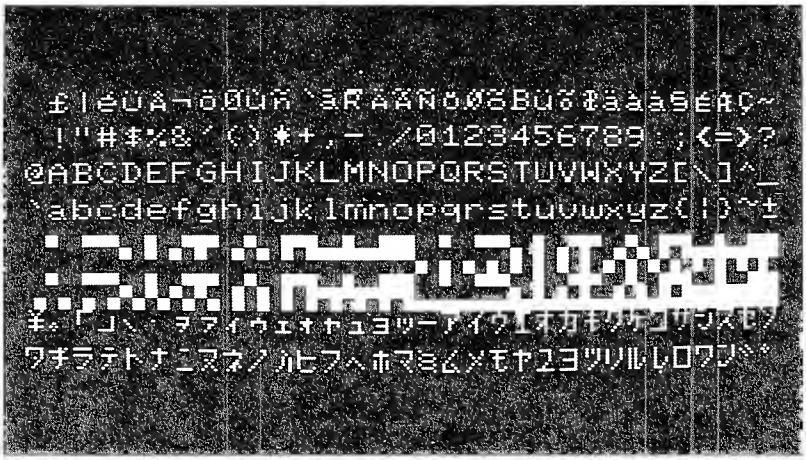


Fig. 1-6. Character set available with bit 3 port ECH low on TRS-80 Model III.

most significant bit position is not used. Instead, these characters are displayed by outputting the hexadecimal values 80H (a graphic blank) through BFH (full white). The least significant bits control the 64 patterns identical to the patterns formed in the Model I system. The enhanced or extended size is also available, but the ports used to enable this function and test the mode activation flag are changed.

Keyboard Input

Only one change is made on the keyboard logic, and it affects the data available on the data lines when the location 3880H is addressed. The right shift key data appears on bit 0, and the left shift data appears on bit 1, instead of both on bit 0 as in the Model I system. As a result, there are additional decoding capabilities and more control characters possible on the Model III system.

Fig. 1-7. Character set available with bit 3 port ECH high on TRS-80 Model III.

Tape Input/Output

Cassette tape input and output is controlled by port FFH. The bits used for reading and writing are the same as those used in the Model I system, but the remaining bits of port FFH do not provide the same functions. The cassette is capable of operating at a data rate of 1500 b/sec using the improved hardware support on the Model III.

Interrupts

Hardware has been included to allow the use of all the interrupt features of the Z-80 processor. It is not necessary to use this hardware. The interrupts are possible using all three modes (0, 1, and 2) without any hardware modifications.

MODIFICATION OF THE TRS-80

We have two methods of modifying the TRS-80 into a development system.

The first involves only software. By simply loading the monitor from tape into available read-write memory and using the SYSTEM command available on Level II machines, all commands in the FROLIC monitor are available. This modification is only possible for the Level II machines, since no method exists for setting breakpoints in a Level I unit.

The second method involves the replacement of the BASIC ROMs.

This replacement means the loss of the BASIC programming feature. These ROMs are replaced with a single programmed EPROM containing the FROLIC monitor. Because of the hardware differences between the Model I and Model III systems, these EPROMS are not interchangeable. For a successful modification it is necessary to have the correct EPROM. The source codes of the two variations of the monitor are listed in Appendixes D and E. To replace the ROMs with the single EPROM, the seal on the unit must be broken. Because the warranty is now voided, there will be an additional charge levied by Radio Shack if service is required on the system.

To remove the BASIC ROMs, the following procedure must be followed. It is necessary that a 2716 EPROM be available preprogrammed with the correct version of the monitor. This EPROM may be purchased from the supplier listed in Appendix C or programmed from the source listings given. An EPROM programmer facility is available with the new monitor, but it is not available until the modification is made. If one starts with a cassette containing the monitor and builds the EPROM programmer described in Chapter 4, the task is simplified. With the programmer, burn in a clean 2716 EPROM with the object code for the Model I system. You now have the monitor in EPROM ready for insertion. This EPROM must be the single, +5 volt variety. It is important that you program the correct program for the model you are modifying. If you are modifying a Model III, go to the section that deals with the modification of that system.

Modification of the Model I

To modify the TRS-80 Model I, the removal and replacement of the BASIC ROMs is necessary. The TRS-80 keyboard must be opened. To do this, place the keyboard unit face down and remove the six Phillips type screws located on the underside of the keyboard. A metalized paper seal may be hiding one of these screws, so be sure you have removed six of them. After removing the screws, carefully place the keyboard unit face up, and then lift the top cover, exposing the bare keyboard. With extreme caution, so as not to damage the flexible connector between the keyboard and the CPU board, lift the keyboard in a manner somewhat similar to the opening of a book. Unfold the board toward you with its bound axis parallel to your chest. Fig. 1-8 shows the exposed computer board. You should see five soft plastic spacers that must be lifted while balancing the keys in one hand. Be careful not to lift the bottom plastic protective cover, as this may place undue strain on the very sensitive connection between the two printed circuit boards. After the plastic spacers have been removed, carefully lift the CPU board

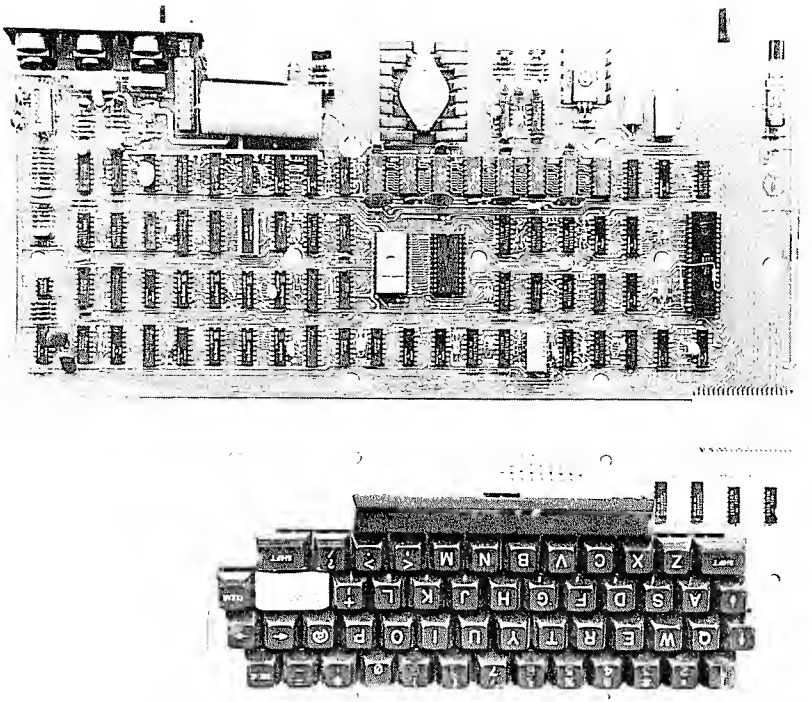


Fig. 1-8. Circuit board layout for TRS-80 Model I.

from the bottom plastic protective cover. Once again try not to lift the entire unit because of the strain this imposes on the interconnection. After successful separation, lay the keyboard and CPU flat on a protective, nonconductive surface. I have spent many hours operating the computer in this position, and with reasonable care no damage will come to the unit.

In a Level I unit and a late model Level II, there is no umbilical cable to the electronics supporting the BASIC ROMs. It is assumed you have such a machine in front of you. A following paragraph will describe the procedure for those units with attached ROMs. Simply remove the two ROMs located in the center of the unit. They are identified by the small letters Z-33 and Z-34, most likely above and to the right of the ROM sockets. Fig. 1-9 shows an enlarged view of the two ROM sockets with the FROLIC monitor EPROM in place. Note carefully the position of pin 1 on the ROM chips. (Pin 1 is on the end of the chip that is notched or marked in some other way. As shown in Fig. 1-9, pin 1 is always to the left of the notch.) Also, make a careful note of the position of the ROM. Mark it if necessary on the printed circuit board using a grease pencil, an

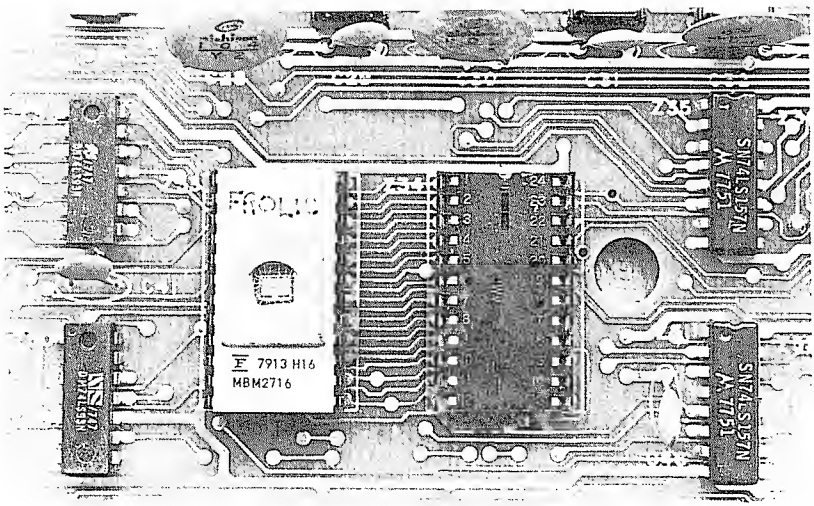


Fig. 1-9. Detail of ROM placement in a Model I TRS-80.

adhesive label, or any other convenient, nonconductive method. The ROMs are generally tight in the socket, so remove them with great care by prying up evenly on both ends.

If the unit you are modifying is a Level I, the new 2716 EPROMs can be placed in either socket position and still operate correctly. If you are modifying a Level II, then place the preprogrammed 2716 in socket Z-33. Since this application only requires one EPROM, no conflict exists in the memory space. However, it is not possible to have two 2716s reside in both sockets at the same time and have the memory space contiguous. That is, there must be a 2K break between the two memories. The select logic for the BASIC ROMs along with the general decoding logic is the final topic of this chapter. Let it suffice to state that if one desires to use the 4K or 8K total ROM space available in one socket, it is recommended that a 2732 EPROM or 2764 EPROM (single voltage) that occupies only one socket be used. Other possibilities exist and will be covered.

After making sure the EPROM is properly installed, it is possible to test the unit by attaching all the connecting cables. Be very careful, since connector orientations are different from what they are in the closed operating condition. The DIN plug closest to the center of the board is used for the power supply. The center DIN plug connects the monitor, and the DIN plug at the outer edge is used for the cassette player. All should test successfully with the prompt "FROLIC:" appearing at the bottom extreme left of the screen. If

this does not occur, replace the BASIC ROMs and check the unit. If the unit operates correctly, the EPROM containing the monitor is at fault, in which case it must be reprogrammed or replaced.

The modification of a Level II machine with the slave memory is almost identical to the procedure used in altering the Level I unit. In these earlier units, there is an umbilical plug connecting one of the ROM sockets to the supporting electronics containing the Level II BASIC ROMs. In addition, there are some extra wires used for addressing. These will not cause trouble when the umbilical plug is removed from its socket. Protect the plug on the end of the cable by any convenient method. Some electrical adhesive tape may be used to achieve isolation of this connection. Proceed as in the Level I modification by inserting the programmed EPROM into socket Z-33 of the two available sockets, and test the system.

Interrupt and Halt Modifications—A third method of modification is merely an extension of the second method in which the mode 1 and mode 2 interrupt feature, as well as the HALT feature, of the Z-80 microprocessor is enabled. In order to obtain this flexibility, the Z-80 microprocessor chip (the only 40-pin device in the keyboard unit, labeled Z-40) must be removed and altered.

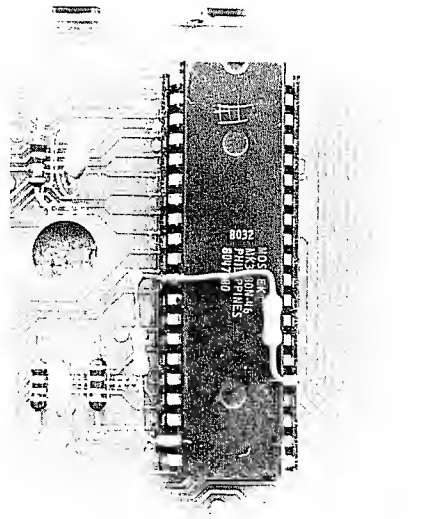


Fig. 1-10. Z-80 modifications for halt and interrupt features.

If you wish to obtain the full interrupt feature, remove the Z-80 microprocessor from its socket. The BUSRQ/ pin (number 25) must be bent outward with great care. Between pin 25 and pin 11, the +5 V dc line, you must attach a 4700 ohm, ¼ watt, 5% tolerance pull-up resistor. Fig. I-10 shows this modification. With this altera-

tion complete, all interrupt features of the Z-80 microprocessor are now available.

As mentioned earlier in this chapter, the HALT mnemonic is not available to the user in the unaltered TRS-80. The designers of the TRS-80 chose to take the halt acknowledge signal (HALT/) and gate it into the nonmaskable interrupt (NMI/) input. To enable the use of the HALT instruction, this line must be removed. The alteration is possible using the same technique used for engaging the full interrupt feature. In this case, microprocessor pin 18, HALT/, is bent outward. Between pin 18 and pin 11, the +5 V dc line, attach a 4700 ohm, ¼ watt, 5% tolerance resistor. This leaves a floating input to Z53 (74LS132), a quad 2-input NAND Schmitt trigger. There is no problem with the floating input, since the other side of that gate is tied to the +5 V supply. If this makes you uncomfortable, tie pins 1 and 2 of Z53 together. The two modifications described give the user complete control in the implementation of all the Z-80 features.

Note that the original Z-80 could be saved and a second Z-80 microprocessor chip could be altered using the pull-up resistors and pin 11 for sourcing the 5 V dc supply. If one is careful with his soldering procedure, no damage will be inflicted upon the microprocessor. Fig. 1-10 shows a possible arrangement and sequence of attaching these pull-up resistors. If repair is required to the TRS-80, the original Z-80 may be inserted into the 40-pin socket, and there will be no visible evidence of any alteration except the obvious broken seal on the protective cover.

Memory Decode Logic—If all the alterations described have been completed, the transformation of the TRS-80 to a development system is now complete. However, some words are necessary about memory expansion to this basic unit if expansion to the full 16K is desired. An understanding of the operation of the dynamic RAMs used in the TRS-80 units is necessary to aid in your modifications to the memory system.

First, consider the hardware conveniences supplied with Model I, Level I and Level II units that overlap both ROM and RAM addressing capabilities. There are two DIP shunt sockets, X-3 and X-71, located inside the keyboard unit. (They may be labeled Z-3 and Z-71 on the circuit board.) It was suggested earlier that you replace these shunts with DIP switches. See Fig. 1-11, which shows these two switches relative to the Z-80 microprocessor. Through various switch combinations, the memory may be configured in a variety of combinations.

The memory mapping of the lower 32K (the total available memory space within the keyboard unit) is controlled by the two DIP

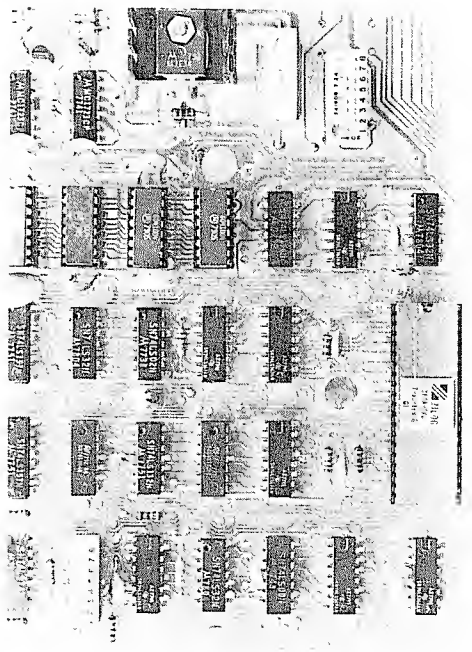


Fig. 1-11. Switch socket positions of DIP shunts X-3 and X-71.

shunts shown in Fig. 1-11. DIP shunt X-3 is located to the right of the memory near the reset button. DIP shunt X-71 is located nearest the ROM sockets. The distinctive style of these packages makes it impossible to overlook them.

The function of X-3 is address decoding. The 32K of memory is segmented into 4K blocks through the use of a dual 2-input to 4-line decoder/demultiplexer (74LS156). This device is configured to function as a 1-of-8 decoder of address lines A14, A13, and A12. Address line 15 is used to select this chip when the computer is accessing the lower 32K of memory. The RAS/ (row-address strobe, active-low), which is in fact the MREQ/ (memory-request signal, active-low) from the Z-80 microprocessor, also is used to activate address selection. Fig. 1-12 shows the significant features of the address select decoder. The exact diagram is obtainable from the TRS-80 technical manual, which is needlessly complex for the presentation here.

The 32K block can be subdivided further into the upper 16K block and the lower 16K block. Address selection for the upper 16K block is straightforward. When the appropriate connections are made on

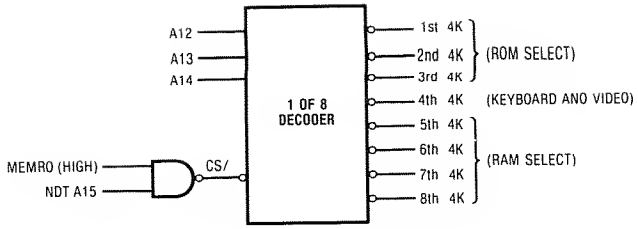


Fig. 1-12. Address selection of lower 32K memory.

the DIP shunt, 4K blocks are selected to activate the MEM/ line which enables the memory present in the sockets to be available to the Z-80 microprocessor data lines. Three types of dynamic RAM devices can occupy the eight sockets: 4K × 1 RAMs, 8K × 1 RAMs, or 16K × 1 RAMs. Selection of the amount of RAM is controlled by DIP shunt X-3, while the shunt at X-71 determines the size of dynamic RAM which may be placed in the memory sockets. Table 1-4 shows the connections to be made to determine how the upper 16K of RAM is accessed.

Read-Only Memory Expansion—The lower 16K of address decoding is not as simple, since there is a great deal of flexibility over the use of this space. Only three of the four 4K blocks are available at the DIP shunt. The missing block address space 3000H to 3FFFH is not present. You should recall that this block of memory contains the video and keyboard addresses. Even though it would be nice to place a monitor in part of this unused space, it would require special decoding to accomplish this. The other three 4K blocks are the first 12K of ROM that is used to support the Level II BASIC, or the 4K Level I BASIC. Because this ROM has been made available in a variety of IC packages, Radio Shack provided some flexibility in the use of the lower 12K. DIP shunt X-3 also controls how the two ROM sockets are addressed, since these sockets may hold 1K, 2K, 4K, and 8K ROM devices. For convenience, socket Z-33 is called ROM A, and socket Z-34 is called ROM B. DIP shunt X-3 now takes on the significance shown in Table 1-5 in addressing this memory.

Table 1-4. Shunt Connections for X-3 to Allow for 4K R/W Memory Selection

From Pin	To Pin	Addresses	
2	15	1st 4K	4000H to 4FFFH
3	14	2nd 4K	5000H to 5FFFH
4	13	3rd 4K	6000H to 6FFFH
5	15	4th 4K	7000H to 7FFFH

Table 1-5. Shunt Connections for X-3 to Allow for 4K ROM Memory Selection

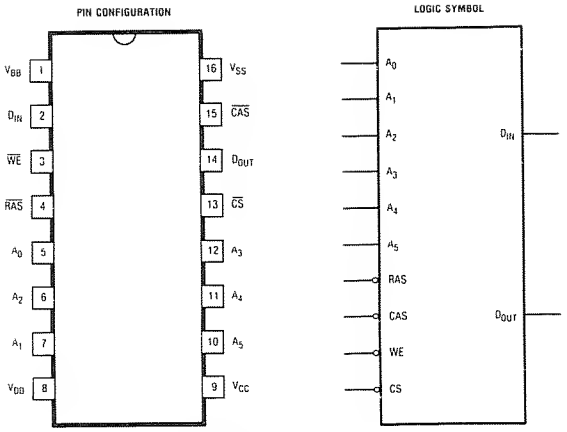
From Pin	To Pin	Address
1	16	1st 4K ROM B 0000H to 0FFFH
7	10	1st 4K ROM A 0000H to 0FFFH
8	9	2nd 4K ROM A 1000H to 1FFFH
6	11	3rd 4K ROM B 2000H to 2FFFH

Notice that a 2K ROM could be plugged into socket Z-33 and Z-34 as is done on some Level I units. Logic internal to the ROM itself used for address selection prevents conflict in this address space and is independent of which socket holds the 0000H address as long as shunt strapping allows access to the correct location. This is accomplished in the Level I units by strapping pins 1 to 16 and 7 to 10 at shunt X-3.

It is also possible to use a single 4K ROM, and again it could be placed in either socket. In both cases the DIP shunt connections between pins 1 and 16 and pins 7 and 10 are shorted. Only one such device is allowed if the shunt is so configured. One can be selective as to which socket is used by controlling the DIP shunt strapping. This is the preferred method of ROM selection.

An 8K ROM can be placed in Z-33 with the appropriate closures between pins 7 and 10 and pins 8 and 9 made at the X-3 DIP socket. However, if the 8K ROM is placed in Z-34 (the adjacent ROM socket holder), the switch does not provide for all the necessary closures using the DIP shunt. By using jumpers to make connections between pins 1 and 16 and between pins 9 and 11 (this is necessary, for both pins are on the same side of the dual-in-line configuration), one can address the first 8K of ROM using socket Z-34. It is interesting to note that Radio Shack originally found it to be less expensive to add the external ROM printed circuit board for the 12K BASIC ROMs than to manufacture 8K and 4K ROMs for use in these on-board sockets. The addressing capabilities are there to use. The potential provided in this home computer by making use of customized ROMs is exciting and worth expanding.

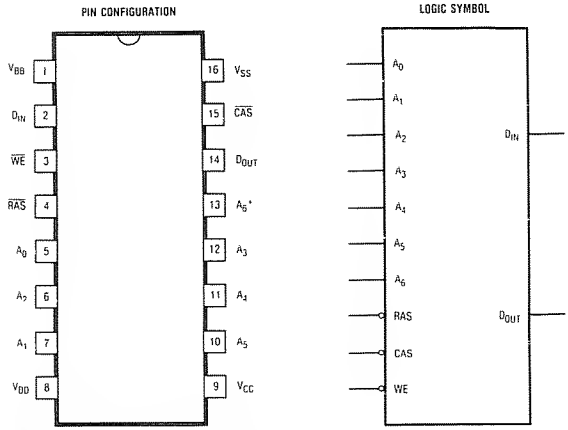
Read/Write Memory Expansion—The control over addressing the dynamic RAMs that are compatible with the TRS-80 keyboard unit is provided by the DIP shunt in position X-71. To understand its function it is necessary to look at the dynamic RAM pinout of the series suitable to operate in these sockets, as shown in Fig. 1-13. Two types of RAM can be used in this unit. They are 1 bit wide, and hence eight devices are required for the addressed word. The "length" of dynamic RAM installed can be 4K or 16K. Although it



PIN NAMES

A_0-A_5	ADDRESS INPUTS	$\overline{WE}$	WRITE ENABLE
$\overline{CAS}$	COLUMN ADDRESS STROBE	V_{BB}	POWER (−5V)
$\overline{CS}$	CHIP SELECT	V_{CC}	POWER (+5V)
D_{IN}	DATA IN	V_{DD}	POWER (+12V)
D_{OUT}	DATA OUT	V_{SS}	GROUND
$\overline{RAS}$	ROW ADDRESS STROBE		

(A) 2104A 4K × 1.



*Note S6000, S6002 A_6 at V_{IH} during row address valid
S6001, S6003 A_6 at V_{IL} during row address valid.

PIN NAMES

A_0-A_6	ADDRESS INPUTS	$\overline{WE}$	WRITE ENABLE
$\overline{CAS}$	COLUMN ADDRESS STROBE	V_{BB}	POWER (−5V)
D_{IN}	DATA IN	V_{CC}	POWER (+5V)
D_{OUT}	DATA OUT	V_{DD}	POWER (+12V)
$\overline{RAS}$	ROW ADDRESS STROBE	V_{SS}	GROUND

(B) 2109 8K × 1.

Fig. 1-13. Logic symbols and pin configurations for

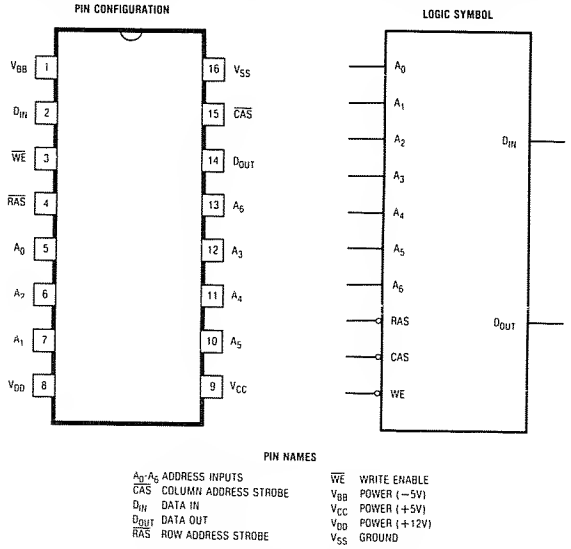
is possible to use 8K dynamic RAM, there is a problem with the installation of these memories because two versions of this memory are available. Details for installation of all of the memories are given. Notice in Fig. 1-13 that only pin 13 shows a change. If 4K devices are used, then pin 13 functions as a chip select (CS/). For 8K devices this pin functions as an address input and chip enable. With 16K devices the chip enable availability is lost, but no serious problem results. Because of these few differences, each memory group size is considered separately. The DIP shunt arrangements given for each group can be seen in Table 1-6.

Table 1-6. Dip Shunt Connections on X-71 for Dynamic R/W Memory Selection

(Check X-3 for absolute address and range.)

4K RAM	16K RAM	8K RAM (Lower)	8K RAM (Upper)
2–15	1–16	1–16	1–16
4–13	3–14	3–14	3–14
6–11	5–12	5–12	5–12
		8–9	7–10

Because of the large address space accessed, it would not be possible to address 4K of memory with only six address inputs.



(C) 2117 16K × 1.

dynomic RAM memory in the TRS-80 microcomputer.

Hence, multiplexing the address lines using the RAS/ (row address strobe, active low) and CAS/ (column address strobe, active low) allows 12 address inputs and consequently the ability to address 4096 memory locations. The address bit assignments A0, A1, A2, etc., are arbitrary, and any order of addresses placed on these pins would result in a unique location being accessed. Multiplexing a 12-bit address to 6 lines is accomplished with two quad 2-to-1 multiplexers (74LS157). Arrangement of the shunts at X-71 determines the size of memory addressed. In this case, the addresses A0, A1, A2, A3, A4, and A5 are strobed during the RAS/. During the column address strobe (CAS/), A6 is directed to the A0 input, A7 to the A1 input, A8 to the A2 input, A9 to the A3 input, A10 to the A4 input, and A11 to the A5 input. The CE/ (chip enable, active low) is made available from RAM/ (memory decode, active low). DIP shunt X-3 selects the appropriate 4K block. In most systems, including the monitor, this is the first 4K block from 4000H to 4FFFH. However, any 4K block could be selected as was shown using DIP shunt X-3 in Table 1-4. The strappings for DIP shunt X-71 for dynamic memory selection are shown in Table 1-6.

Memory addressing for the 16K devices replaces the CE/ line with A6. With this addition of another pin for addressing, 14 address bits can be strobed into the chip memory latches, which results in 16.384 locations being accessed. The CE/ function for these memories is combined with the RAS/. The additional external hardware to provide for the decoding is minor when compared to the increased memory capacity. The DIP shunts at X-71 must be changed to reflect the insertion of the larger memory chips. The new positions are shown in Table 1-6. By installing 16K RAMs into the TRS-80, the DIP shunt routes address line A6 to the A6 input (formerly the CE/ of the 4K memories). This address is strobed along with the other addresses during the RAS/ activation. However, during the CAS/ activation A13 is directed to A0, and A12 to A6. All other addresses, A7 through A11, remain the same as those strobed with the 4K devices during the CAS/ activation. An examination of Table 1-6 should help to clarify the addressing scheme for each of these different types of RAMs.

The 8K dynamic RAMs present another variation for the A0 input. In this case the A0 line is used to strobe the operating half of a 16K dynamic RAM. It is not clear why this method of 8K implementation is used. Intel Corporation supplies an 8K dynamic RAM that uses the A6 input during the CAS/ to function as a chip enable. Two versions are available, one in which the memory device is enabled with this line input low, and the other when this line input is high. Thus the chip enable feature is still provided for in the 8K device. If A6 is

low, the lower 8K is addressed; if it is high, the upper 8K is addressed. The unique location is determined by the logic levels strobed with the other address lines. (It appears that these devices are fallouts from the manufacture of 16K memory.) DIP shunt X-71 makes provision for a logic high/low not for pin A6 but for A0. It seems that Radio Shack was going to have an 8K RAM manufactured with this specification. One assumes that because of the low cost and availability of 16K memory, making customized versions of 8K devices became prohibitively expensive, and the scheme was discarded. If an 8K dynamic RAM system is desired, this can be accomplished through the use of the 16K memories. This will allow 8K of additional ROM within the keyboard unit. You must supply your own hardware to decode the ROM address space. DIP shunts on X-71 provide for either the low or high addresses on A13. Memory not to be accessed may be isolated with DIP shunt X-3. Few systems have been modified to use 8K of RAM. However, the electronics for such a conversion is present, in this unorthodox fashion. Table 1-6 lists the DIP shunt selections possible for an 8K implementation. Do not forget to make the appropriate range selections with X-3.

To clarify further the address data on the dynamic RAM pins during the RAS/ and CAS/, Table 1-7 is provided.

Modification of the Model III

To modify the Model III, the ROMs must be removed from the desk-top unit. To accomplish this, rest the unit on its back or top, protecting the surface with a soft cloth or foam padding. There are 10 Phillips screws surrounding the outside of the Model III. Do not remove the screws in the plastic feet. The function of the feet is to support the unit and prevent marring. They do not retain the cover and will have no effect on the disassembly. Remove the 10 screws at the outer edge. You will notice that these screws have different thread styles and lengths. The three shortest have machine threads and are in the front holding that portion of the case that covers the keyboard. Behind these, toward the edge and in line with the front of the crt structure, are two longer machine-thread screws. These screws are in a well about ¾ inch deep. The remaining five screws are 1-inch sheet-metal screws. One of these screws may be covered with the seal that refers to the warranties. The breaking of the seal will result in an additional service charge if the unit is to be returned for repair. (If this unfortunate event should occur, do not forget to restore the original ROMs in the unit.) After the 10 screws have been removed, do not lift the base because there is one more retaining screw holding the cover in place. The final screw that must

Table 1-7. Pin Assignments and Signals Present on Dynamic R/W Memory

Pin Assignment	RAS/	CAS/ 4K	RAS/ 8K & 16K	CAS/ 8K	CAS/ 16K
5 A0	A0	A6	A0	G or +5	A13
7 A1	A1	A7	A1	A7	A7
6 A2	A2	A8	A2	A8	A8
12 A3	A3	A9	A3	A9	A9
11 A4	A4	A10	A4	A10	A10
10 A5	A5	A11	A5	A11	A11
13 CE/ or A6	RAM/	RAM/	A6	A12	A12
1 Vbb					
2 DIN					
3 WE/					
4 RAS/					
8 Vdd					
9 Vcc					
14 DOUT					
15 CAS/					
16 Vss					

be removed is in the top center of the back of the unit. In the inverted position, this screw is in the center back near the resting surface. Remove this last screw, and carefully lift the base from the top cover. The top cover holds the crt and associated electronics and is attached to the main base with a long connecting cord. Lift the base vertically high enough to clear the circuit board from the neck of the crt and the cover. The base is light, so with reasonable caution no problem will occur. After you are certain of being clear of the top cover, rotate the base by 180 degrees and set it on its plastic feet. The removal of the BASIC ROMs can now take place.

Facing the unit from the rear, you will notice in the lower right-hand corner three large 24-pin dual in-line integrated packages (refer to Fig. 1-14). These are the three ROMs that contain the Level II BASIC. If the Model III is a Level 1, there will be only one socket of the three filled, the extreme left socket. Fig. 1-15 shows an enlarged view of the ROM sockets with the FROLIC monitor in place. These sockets are labeled U104, U105, and U106. The socket closest to the cassette plug is U106, the system utility ROM. The memory capacity of this IC is 2K × 8. The center socket, U105, can hold a 4K × 8 ROM. The left socket is U104, and it resides at the starting address, 0000H. In a Level 1 system the size of this ROM is 4K × 8, and in a Level II system it is 8K × 8. It is this ROM that must be removed to reconfigure the TRS-80 into the FROLIC development system. Therefore, remove it, but note the position of pin 1, which is in the top left corner of the package. The ROM will

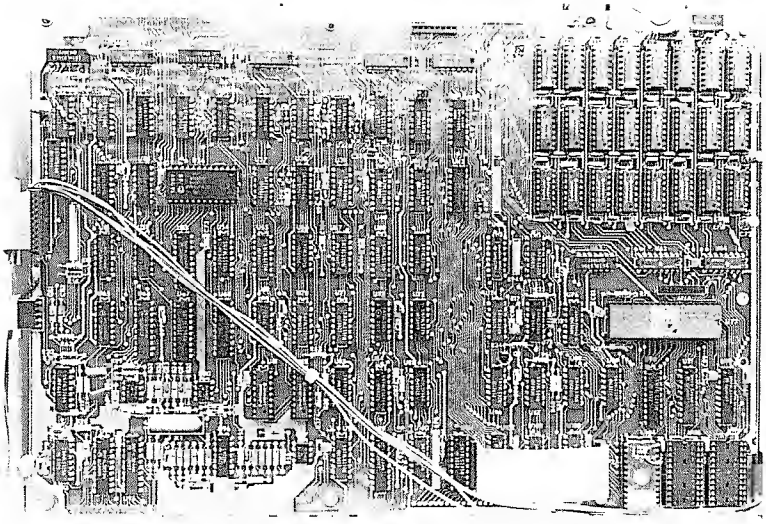


Fig. 1-14. Circuit board layout for TRS-80 Model III.

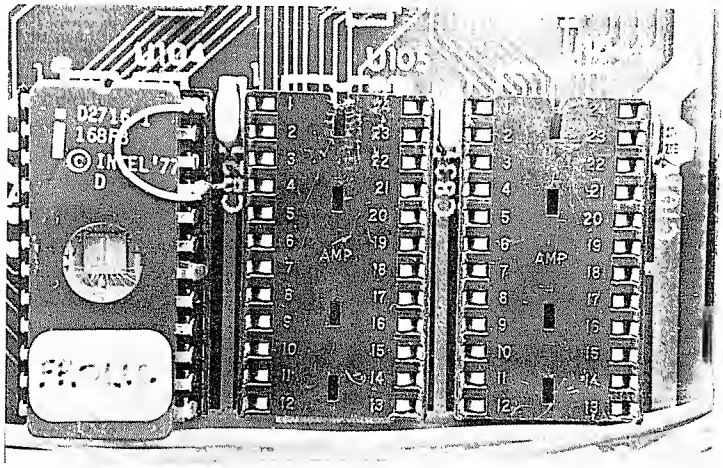


Fig. 1-15. Detail of ROM placement in a TRS-80 Model III.

be tightly pressed into its socket, so remove it with great care using a small flat-blade screwdriver which allows you to get between the ROM and the socket for greater leverage. After the ROM has been removed, you are ready to replace it with a 2716 EPROM obtained from the supplier listed in Appendix C or with a 2716 EPROM that you have programmed using the object code given for the Model III

in Appendix E. Pin 1 is inserted in the upper left of socket U104. However, with a 2716 EPROM, one precaution must be observed. It is necessary to raise pin 21 to +5 V dc to enable the chip at addresses that start with location 0000H. This is accomplished by making a connection between pin 24 (the V_{CC}) and pin 21. This can readily be observed in Fig. 1-15. The ROM supplied from the listed source has this jumper in place. Pin 21 must be (or is) bent slightly outward so that it will not be pressed into the socket when it is inserted. It is possible to place either a 2732 (4K × 8) or a 2764 (8K × 8) EPROM in this socket. The 2732 and 2764 do not require a jumper wire. (A note of caution concerning the 2716. If it is necessary to reprogram this EPROM, do not attempt to program the 2716 with this jumper in place.)

The Model III may be tested with the cover off. After the EPROM is in place, place the crt screen in a position so you can see it. Turn on the unit with the switch under the keyboard on the right. The "FROLIC:" prompt should appear on the lower left of the screen. If not, replace the original ROM, and try the unmodified system. If all is well, as it should be, the fault is in the EPROM. Return it to the supplier, or check the object code against the listings for the Model III unit given in Appendix E.

Unlike the Model I, full memory expansion is provided for on the microcomputer circuit board. In the upper right corner are sockets for 24 dynamic R/W memory devices. The first row may take eight of either the 4K × 1 or 16K × 1 dynamic RAM devices. If the unit

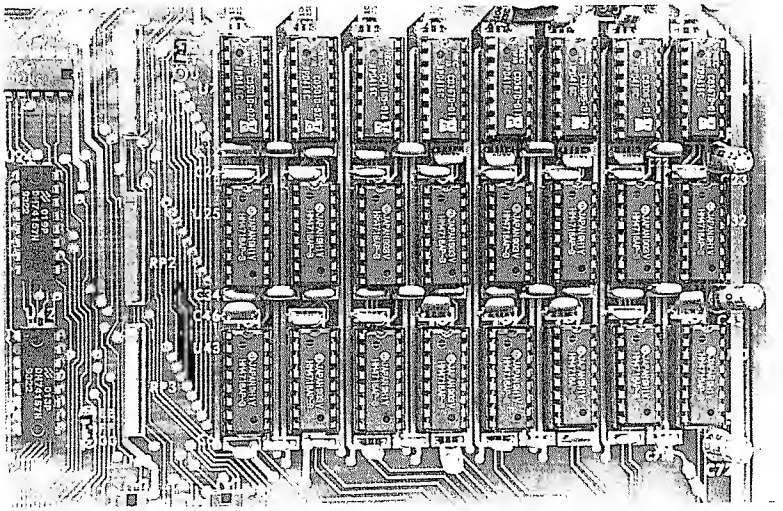


Fig. 1-16. Dynamic R/W memory placement in the TRS-80 Model III.

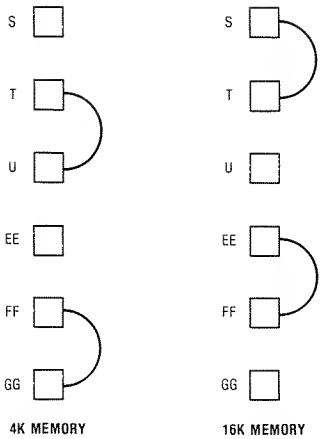


Fig. 1-17. Jumper placement for 4K and 16K memory selection in TRS-80 Model III.

is a Model III Level I, the 4K devices are residing in the upper row, with all other socket holders empty. To convert to a 16K machine, it is necessary to change the address select and chip enable of this first row. This is accomplished by special jumpers to the left of the memory. Refer to Fig. 1-16, in which the jumpers are shown near socket U-7 and near RP-3. The upper jumper is labeled, from top to bottom, S, T, and U. The lower jumper is labeled EE, FF, and GG. For 4K memory, T is jumpered to U with S open, and FF is jumpered to GG with EE open. For 16K memory, S is jumpered to T with U open, and EE is jumpered to FF with GG open. Fig. 1-17 shows the placement of the jumpers for the two types of memory that may occupy the first row.

The memory can be expanded to the full 48K capability by filling the remaining 16 RAM sockets with 4116 dynamic memories. The jumpers to the left of memory should be in the 16K position as shown in Fig. 1-16.

CONCLUSION

The information provided in this chapter is to aid in understanding the TRS-80 system. To some readers the material may appear too complex, and to others not complex enough. Attention was given to memory utilization, input through the keyboard, output to the crt, and tape storage techniques. The software provided requires some knowledge of these system components. It is also hoped that the potential for this low-cost microprocessor development system is realized. Its uses may be far removed from those envisioned by the TRS-80 system designers.

The intent of this book is to provide the software to obtain a sophisticated monitor operable on the TRS-80 system. The software is given in the Appendixes. Chapter 2 describes the command set of this monitor.

CHAPTER 2

The Monitor

The FROLIC monitor transforms the TRS-80 microcomputer into a sophisticated and manageable development system. The monitor allows the user to interact directly with the TRS-80 at the machine-language level, to enter and execute machine-language programs, to read and write machine language tapes using audio cassettes and a recorder, to use editing and debugging facilities, and to program several of the currently popular and readily available programmable read-only memories (PROMs).

The monitor is intended to replace the first 2048 (2K) bytes of read-only memory currently residing in any of the TRS-80 Model III and Model I microcomputers. An EPROM is used for this replacement. Alternatively, the monitor program may be entered into memory using a cassette as presented in Chapter 1. The advantage of ROM replacement is that this allows instant availability of the monitor on power-up.

COMMAND FORMAT

Most commands are designed to be single-character commands. Up to three hexadecimal addresses may follow a command. In this text the hexadecimal representation is always indicated by the presence of a terminating upper case H with the only valid address digits being 0 through 9 and A through F. Likewise, decimal addresses terminate with an upper-case D with the only valid entries being the digits 0 through 9, and binary addresses terminate with an upper-case B with the only valid entries being 0 or 1. Upon entering a command, addresses and address data are separated by spaces with the

first space being optional. The letter "T" is used to separate addresses, as though the "T" represents "through." The command formats that represent typical command entries into the monitor are shown in Table 2-1.

Table 2-1. Typical Monitor Command Instructions

Command	Definition
C	No address required
D40	0040H is the only address affected
D4000	4000H is the only address affected
D4000T4040	4000H through 4040H is affected
D4000 40	40H or 64D positions are affected, an alternate form of the preceding
M4000T4040 5000	Three fields are the maximum
M4000 40 5000	An alternate form of preceding format
D934000 564040	Equivalent to D4000T4040 or D4000 4040

Note that:

A. Commands may not need addresses.
B. Leading zeros are optional.
C. Leading spaces are optional, and only the first occurrence is used for separation of address or data fields.
D. If more than four hexadecimal numbers are entered together, only the last four are recognized.
E. If the T delimiter is not used, spaces may be used instead with the first address indicating the initial address and the second hexadecimal value representing the number of positions affected, first address inclusive.
F. If entered addresses or address data is separated by a letter other than T or a space, the command is ignored.

COMMAND EXECUTION

All commands are executed by depressing the <ENTER> key. Editing of a command or command string is done through the use of the left arrow key for backspacing, which allows you to "back up" to the error for correction. Command editing may be continued until <ENTER> is finally depressed.

The monitor for the TRS-80 is described in the text that follows. At the end of the text, documentation is included to enable you to modify any of the subroutines to fit your particular purposes. Expansion to 26 commands is possible, one for each letter of the alphabet. In addition, the "at" @ is included, for a total of 27 commands. Most commands utilize a letter that is close to describing the desired com-

mand. For example, to display a section of memory the upper-case D is used, indicating "Display." Also, depending upon the command, up to three data fields are required. Certain formats are used to specify ranges of addresses that occur in these commands. A great deal of effort was made to be consistent.

Monitor operation is indicated by the initial "FROLIC:" prompt. This "sign-on" appears in the lower left area of the screen, at the beginning of the sixteenth line. If this is not the case, resetting the CPU using the reset button should result in the display of the prompt.

Table 2-2 shows a list of the 27 possible commands. Each command is described in the text that follows.

Table 2-2. List of Available Commands in the FROLIC Monitor

Command	Definition
@	Fill memory with constant
A	ASCII display of memory specified in command
B	Repeat last command saved in buffer
B(S)	Save command string and execute with a B command
C	Clear screen
D	Display memory in hexadecimal format in range specified
E	Execute with current screen data
F	Used for searching for a single byte of data over the range specified
G	Go and execute the program starting at specified address
H	Hexadecimal sum and difference
I	Insert data from command string
J	Not used
K	Not used
L	Produce hard copy on listing device (if this option is loaded)*
M	Move memory data from one location to another
N	Input data from input port(s)
O	Output data to port specified
P	Program PROM (if option loaded)*
Q	Search memory for a two-byte word in INTEL format, low byte followed by high byte
R	Read cassette tape using Radio Shack 500 bits/second format
S	Substitute to memory from keyboard input
T	Transfer data from previously allocated screen storage to crt display
U	User assignable*
V	Verify memory
W	Write a cassette tape using Radio Shack 500 bits/second format
X	Examine and alter Z-80 registers
Y	Not used
Z	Single-step (hardware must be implemented)*

*The implementation of these commands is described in text.

Commands are executed from a buffer. When any key is pressed, its ASCII (American Standard Code for Information Interchange) value will be stored in this buffer. Multiple commands may be entered by separating each command by a comma. The buffer is located starting at address 4080H and ending at address 417FH.

For proper command execution, the correct format must be entered. If you fail to enter the correct format, execution is attempted, but it will cease when an unexpected character is encountered. At this time, the complete command buffer is displayed with a "?" over the nonexecutable entry. If possible, any additional commands that remain in the command buffer will be executed; otherwise execution stops. The prompt reappears, and the correct format of the command must be entered. Editing of the line is possible until <ENTER> is depressed. You can "escape" from the command sequence executing by actuating the <BREAK> key. Then the computer will indicate that the monitor is ready to accept a new command.

To use the monitor, it is necessary to specify a command and where necessary specify the data fields describing the extent of operation of the command. Each command has its particular field specification. In some cases similar formats occur in several of the command fields. For example, both the A and D commands require a beginning address and a specification for the number of bytes to be displayed. If an error is made in the entry, it is only necessary to use the backspace arrow to "rub out" the last entry or entries and insert the correction. In any address or address field, only the last four hexadecimal digits are accepted as valid data.

Assume you have entered a command and it is executing, and you wish to halt or abort the command. The process may be halted by pushing down and holding down the space bar. This stops execution for as long as the space bar is depressed. To stop execution completely, simply depress the <BREAK> key. The "FROLIC:" indicates that the monitor is ready for its next instruction.

There are five delimiters or separators used in this monitor. The space " ", the "T", the period ".", the comma ",", and the slash "/". Other characters of the ASCII set used are the <ENTER> or carriage return (CR or 0DH), and start-of-heading (SOH or 01H). The <ENTER> is reserved to indicate to the monitor the command string is to be executed. Hidden from the user's view is 01H, which is used by the monitor to indicate that the last command in the sequence has been executed. The meaning of start-of-heading is important, since the monitor detects the 01H value. The comma "," is used to separate commands. It is the only separator used for this purpose. You may observe this by noting the last entry in the buffer

when an error is encountered, for even though you depressed the <ENTER>, a comma terminates the command string. The period "." is used for the entering of data to be used as breakpoints. The "/" indicates an address change in both the substitute and insert commands. Particulars are covered in upcoming sections that deal with the command descriptions.

Each command is described in detail. These are presented not in alphabetical order, but in a utilization order to allow you to start to use this monitor by knowing only a few commands. As with all programs and systems, usage breeds familiarity. You should practice with a few commands and then add commands as needed. The table in Appendix A places the commands in alphabetical order for quick reference.

S COMMAND

The most flexible of the monitor commands is the S (Substitute) command. This command allows the user to insert data into memory. When using this command an address is required in order to specify where substitution is to take place. The usual format is an S followed by an address. For example:

```
S4400<ENTER>
```

Execution of this command results in the display of the address, 4400H, in the lower-left portion of the screen followed by a colon and the data currently located at that address. A question mark ends the sequence, prompting the user for an entry from the keyboard. The display resulting from the execution of the above command, S4400 <ENTER> is:

```
4400: XX?
```

(where XX is the hexadecimal data stored at 4400H)

You may now enter data. Only valid hexadecimal data is accepted. To use this command, several choices are available. If no change in the data is desired, the user simply depresses the space bar. This advances the memory pointer to the next location. Valid hex data is entered by way of the keyboard, the last two hex digits before depressing the <ENTER> key replacing the memory contents. You may note that if an entry is made which is not hexadecimal, a "?" will appear. Any number of "?" symbols may separate the last two valid entries. It is not necessary to re-enter the command mode to change the address. It is only necessary to enter the new address followed by the "/" in place of the data. The new address location is displayed in the lower left-hand corner. Substitution of data may

now continue as before. It should also be noted that when data is entered into a memory location, that location is then re-read and displayed as an "echo." This technique makes it evident that if data insertion is attempted in ROM or outside of valid read/write memory, the echoed data will not agree with the desired data to be deposited and, therefore, corrective action is required. Additional flexibility is demonstrated in that it is also possible to step backward using the back arrow or "rubout." If, while in the command entry mode a rubout occurs, in this case in the S command, the address is decremented by one and displayed along with the data currently residing at that location. To exit from the substitute command mode the <ENTER> key is depressed twice in succession. It should be pointed out that the double <ENTER> will not alter the contents of the last memory location.

I COMMAND

A second method is available to substitute data into memory using the I (Insert) command. For example:

```
I4400 00 11 22  33 44 4410/ AA BB CC<ENTER>
```

As before, the beginning address of the substitution follows the I command. However, instead of using <ENTER> the user deposits the data in sequence. Each value is separated by a space, the delimiter which is used after each data entry. The sequence continues for as many entries as desired. The execution of the I command shown in the example would result in the data 00H, 11H, 22H, 33H, and 44H inserted in sequence starting at 4400H and ending at 4404H. Notice the next entry (44I0/) is terminated with a "/", which changes the pointer, causing the insertion to continue at the new location. The data AAH, BBH, and CCH is inserted at 4410H through 4412H. After entry of this command, the display appears as follows:

```
4400: 00 11 22 33 44
4410: AA BB CC
```

After the data is entered, you can see the displayed deposits when the command is executed by depressing <ENTER>. It should be noted that the question mark is not used in this command. The display is similar to that obtained with the D command (which follows). Like the S command, these deposits are echoed. If an attempt is made to deposit data in memory other than RAM, the display will reflect it.

D COMMAND

After data is deposited using either the S or I command, you can examine this data through the use of the D (Display) command. A "D" followed by an address with no range specified results in the display of the data in 40H locations. A wider range of data can be observed by using a block format. A block is specified by first using a space as a delimiter after the start address is entered and then following the definition with a value that indicates the block length. For example:

```
D4800 10<ENTER>
```

results in the display of 16 (10H) memory locations. Block lengths up to FFFFH are possible.

An alternative form of this command is to use the beginning address followed by a "T", then the ending address. The result is that the block is specified by the range of the entry. For example, the display commands shown below result in a display of 40H locations beginning with address 4040H.

```
D4040 T 407F<ENTER>
```

or

```
D4040 40<ENTER>
```

Note that the space delimiters as shown on either side of the T are not required. The display that results after executing this command follows:

```
4040: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
4050: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
4060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
4070: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

G COMMAND

After you examine your program in its hexadecimal form and you are convinced that the object code shown should run successfully, execution or testing is the next logical step in program development. This is accomplished with the G (Go) command. The G command requires an address at which execution is to begin. This address can be specified as part of the command, or it can be obtained from the program counter register as saved from a previous execution. In its most simple application, the G followed by an address causes execution to begin at the address specified. You must be as sure as possible that when this command is executed your program will not produce a system "crash," in which case the computer is advancing through

memory in an uncontrolled fashion. A reset will quickly return control to the monitor program, but the contents of RAM, and as a consequence, your program, may have been altered. As an example of the use of the G command, first insert the following object code by using the I command:

```
I4400 00 3E 2F AF C3 00 44<ENTER>
```

The monitor responds with an echo of the object code at these addresses.

```
4400: 00 3E 2F AF C3 00 44
```

This code when executed using the G command does the following: the 00 is a NOP or no-operation (the program counter is advanced), then the computer loads the accumulator with 2FH, clears it (which does affect the flag register), and then does this all over again, and again, and again. It will serve as an excellent demonstration for the G command. With the program in place, execute the following command:

```
G4400<ENTER>
```

Nothing happened, or did it? First you will notice no prompt. Now you should also notice that there is no response to any key entry. What is happening is that the microprocessor is executing code that causes the computer to be in an endless loop. The only choice possible for you at this time is to push the reset. You could use the power-on clear by turning the power off and turning it back on again. However, the program you just entered would be destroyed because of the volatile nature of the dynamic RAM.

Obviously, you would like to have more control over the execution of a program during the program development stage, and a controlled return to the monitor is desirable. This can be accomplished by means of breakpoints. Simply stated, a breakpoint in a program causes execution of your program to cease at a point you have specified, and control is returned to the monitor. When control returns to the monitor, the states of all of the Z-80 registers are saved and then displayed immediately on the crt screen. The condition of the screen may also be saved, as will be shown later. Also displayed on return are the next four bytes which start with the data pointed to by the program counter. This code display occurs only when a breakpoint is encountered. You should also be aware that the code at the breakpoint is not executed. As will be shown later, the state of the registers is always available to the user upon request using the X command. If this request is made, the object codes starting at the breakpoint *ARE NOT* displayed. If the program terminated at a requested

breakpoint, execution can continue by entering "G" followed by a new breakpoint. You may have more than one breakpoint.

To use the breakpoint feature, two alternatives can be employed. If the execution is to start at an address other than that specified by the program counter, the user enters the following type of command:

```
G4400.4401<ENTER>
```

In this case, execution starts at location 4400H, with the breakpoint set at address 4401H. If the short test program used with the G command is still in R/W memory, the computer executes the 00H, or no-operation command, and then reaches the breakpoint. Once the breakpoint has been reached, the monitor responds with the following display. The format of this display is covered in detail in the X command.

```
P=4401 S=4300 X=0000 Y=0000 N=00 I=00 V=3C00
A=00 B=00 C=00 D=00 H=00 L=00 M=0000 F=00
A'00 B'00 C'00 D'00 H'00 L'00 M'0000 F'00
4401: 3E 2F C3 00 44
```

The program counter is now at address 4401H, which was the specified breakpoint. It is assumed that at location 4401H there is an executable instruction. That is, the breakpoint cannot exist within the address field of a call or jump instruction, nor can it be in a position that data would occupy. For example, you cannot execute the following sequence and expect to return to the monitor at the break.

```
G4401.4402<ENTER>
```

The data at 4402H is 2FH, and this is part of the load-accumulator instruction. There exists an instruction in the Z-80 code with a value of 2FH. However, when the load is executed the 2FH is considered data. When a breakpoint is requested, the instruction residing in that location is replaced with an instruction which causes a return to the monitor. Upon such a return, all breakpoints are cleared, so all of the original program steps are restored to their original state, as they were loaded prior to the use of any breakpoints. The only restriction placed on the breakpoint is that it must occupy the position of an instruction.

More than one breakpoint can be specified. In fact, room is provided in R/W storage for at least ten breakpoints. It is not envisioned that any user will require that many breakpoints in checking the operation of a program. For you to implement the multiple breakpoint feature, you simply place your breakpoints in sequential fashion using the period as a delimiter. For example, the following is the format:

```
G4400.4401.4403.4304<ENTER>
```

With this command, all instructions at the three locations specified are replaced with a single-byte call to the monitor. Each instruction is saved in R/W memory along with its address. Execution begins at the specified address and stops at the first breakpoint encountered. All breakpoints are cleared, and the instruction formerly occupying that position is replaced. The breakpoints are entirely "transparent" to the user unless, of course, the user has lost control in his program because a crash has occurred. On rare occasions, this crash may not manifest itself to the user, and unexplained problems may appear. If the program counter does not agree with one of the set breakpoints, an improper termination has occurred. Also, if upon examination of your program, some unexplained FFH codes appear in the program, a crash has occurred. The FFH is used by the monitor to indicate a breakpoint, and it remains if a break is not terminated properly. If the crash is not severe, the monitor will attempt to clear breakpoints upon the depressing of the reset button.

Assuming a proper break is encountered and program execution is to continue at the address of the breakpoint, the user may restart the program by using the G followed immediately by the new breakpoints.

```
G.4403.4404<ENTER>
```

Notice the use of the period to indicate the breakpoint location immediately following the G. Notice also that upon termination from the program by means of a breakpoint the program counter agrees with one of the specified breakpoints.

As indicated earlier, there are two methods of setting the breakpoints. The second method makes use of relative addressing. The addresses are relative to the program counter, and the arithmetic operations of plus (+) and minus (−) are used for breakpoint entry. The value following the operand is added to or subtracted from the program counter to establish the breakpoint. The range of change ("bias") is 0000H to FFFFH. Of course, the use of large relative jumps (biases) can result in what is called "wrap-around"; that is, once the value of the microprocessor's program counter exceeds FFFFH, it wraps around to 0000H without regard to the overflow. An example of the use of the relative breakpoint feature is:

```
G4401.+2.+3.−1<ENTER>
```

Here the breakpoints are the same as if the following were entered.

```
G4401.4403.4404.4400<ENTER>
```

This feature makes breakpoint entry more convenient. Try it with

the example program. The first breakpoint returns control to the monitor with the program counter at 4403H.

Since the register display occurs when a breakpoint is encountered, let us examine in detail what the various registers represent.

X COMMAND

The X (eXamine registers) command is executed by pressing the X key after the prompt appears on the monitor or by making it part of the command buffer. In both cases, the register data displayed is that generated by the user during execution of the test program and not the data generated by use of the Z-80 internal registers by the monitor. The command:

```
X<ENTER>
```

produces the following display when the system is first powered.

```
P=0000 S=4300 X=0000 Y=0000 N=00 I=00 V=3C00
A=00 B=00 C=00 D=00 E=00 H=00 L=00 M=0000 F=00
A'00 B'00 C'00 D'00 E'00 H'00 L'00 M'0000 F'00
```

The first position on the first line is the program counter (P=). In the power-up initialization it is set to 0000H. If a G is entered as a command, the monitor would cause execution of the program beginning at that location. The next register displayed is the user stackpointer. The monitor initializes a reserved area for the user stack. This area is not used for any other purpose and extends from 42C0H to 42FFH. The stack is used in reverse order, and the first position filled with data is one less than that indicated by the stackpointer. The stackpointer always resides at one location above that location used to store the data on the stack. Therefore, data should be stored in the stack beginning at 42FFH. If the stack is not changed by using a stack modification instruction, then this register should be at 4300H. The monitor stack resides at 42C0H and ends at 4280H.

The third and fourth locations contain the information stored in index registers IX and IY.

The next register is the N register, and it can take one of two values. If the value is 00H, then the maskable interrupt is not enabled and no interrupts are possible when control is passed to the user program. If the register contains a 01H, then the interrupt is enabled when control passes to the user program. The I register is the interrupt vector register and has meaning only if the vectored interrupt mode (mode 2) is enabled. This register contains the upper portion (the high-order byte) of the address of the vector. The lower address is available from data strobed on the data lines. This occurs when the interrupt acknowledge (INTAK/) is issued by the

processor after an interrupt is requested. It is possible to cross page boundaries, although the technical manuals seem to imply that this is not the case. More detail of the use of the I register is given in the section dealing with the hardware modification that enables the full interrupt structure of the Z-80 microprocessor.

The last entry in the first line is the location of stored video information. This is a monitor function and not a Z-80 function. The video information consists of 1024 bytes of data that make up the screen. Upon initialization, the video pointer resides at the same location as the beginning of the video screen memory block. The data located at these screen memory positions is stored in those same positions when control returns to the monitor. This data is then scrolled when the register information is displayed. Any writing to addresses in memory that reference the upper video starting at 3C00H and ending at 3CBFH are scrolled and lost forever and cannot be recalled by the user. It is sometimes desirable to see alterations you wish to make to the screen display area of memory without having them scrolled upward when return is made to the monitor. If the V address is changed to point to 1024 locations of selected R/W memory, the data present in the video memory block at breakpoint time is transferred to the location starting at the address indicated. At any time. while under monitor control, this data can be viewed using the T (Transfer) command (the changing of the pointer may be accomplished with the X command as will be shown). Therefore, any data written to the screen while in the user program is readily observable. It should also be noted that before execution of your program the screen data present at the pointer is transferred back to the screen.

The next two lines of the display register command contain the data present in all of the Z-80 registers. All of the registers internal to the Z-80 processor are loaded with the displayed data (in this case 00). This particular pattern is present at power-up as part of the initialization. A return through a proper breakpoint saves the status of these registers at the time of the break. They are displayed from left to right starting with the A (accumulator) register, the B and C registers, the D and E registers, and the H and L registers. The next position is the H and L combined to form M (memory pointer) (added because of the many references made by this pointer). The next position is the F or flag register. This register shows the flags that are set.

```
SZXHX/NC
```

The above F register display is interpreted as sign, zero, not used,

half carry, not used, parity/overflow, subtract, and carry. If no alphabetic representation appears, it implies that no flags are set.

The last two lines are of the same format. The difference is that the first represents the working registers and the second represents the mirror or prime registers. The use of these registers is primarily for servicing interrupts quickly, but they come in handy for many operations that require the use of a register without destroying the previous value in that register. All the registers internal to the Z-80 processor are loaded with the data displayed, and a return to the monitor through a proper breakpoint saves the status of the Z-80 registers at the time of the break.

X COMMAND (MODIFY)

The X command may be used to modify the contents of any of the displayed registers. To use this feature, you enter the new data preceding the <ENTER>. For example, to change the video storage from 3C00H to 4C00H, perform the following:

```
XV4C00H<ENTER>
```

The monitor responds,

```
V=3C00  V=4C00
```

The change remains in effect until the register is altered with the X command, a power reset occurs, or a G0 command is executed. To change any of the other registers the format is the same. Just follow the X with the register name as displayed. The registers are as follows: P (Program counter), S (Stack pointer), X (IX Register), Y (IY Register), N (Interrupt enable upon execution—00 for no and 01 for yes), I (Interrupt high vector address in the Z-80 microprocessor), V (Video storage in RAM), A, B, C, D, E, H, L (the working registers), M (pointer formed by HL), and A′, B′, C′, D′, E′, H′, L′, M′ (mirror registers).

Note that the command,

```
XF' FF<ENTER>
```

produces the display:

```
F'00  F'FF SZXHX/NC
```

All of the registers are alterable with Z-80 object code. However, once the video storage address is changed, only the XV, G0, or power-on clear operations can reset this address. Once located to a valid R/W location, the execution of a G command takes the data at that location and transfers it to the screen RAM prior to execution. You may wonder where the "garbage" comes from when you execute

this command. Returning from a break takes the data in the screen area and saves it at the allocated storage area. This is done so that user programs that require the video as the output device can be developed with greater ease.

A method exists for clearing the screen prior to execution. A second command for execution with breakpoints is included in the monitor command set. This is the E command and is the topic of the next section.

E COMMAND

The E (Execute) command is identical to the G command, except that the current screen information is used at the start of execution. If in the user program any output to memory 3C00H to 3FFFH occurs, the new data replaces the data formerly in the addressed position. This command, in conjunction with the C, T (descriptions of these follow), and XV commands, may be used for scroll protection. When scrolling occurs, the top line of the crt display is erased, and the lines below are all shifted upward by a single line. Essentially, this is a line-feed when the carriage is returned on a typewriter. The FROLIC monitor allows you to protect the display area so that this information is not lost. This is done with the XV command as described. As with the G command, return from execution relocates the screen data to the storage area indicated by the V address. It is the user's responsibility to keep a record of the memory space to avoid conflicts. The format of this command is:

```
E4500<ENTER>
```

Like the G command, execution begins at location 4500H. The difference is that the screen data is not replaced with the data that was in the reserved memory space. Breakpoints may be included as shown:

```
E4500.+1.4510<ENTER>
```

Breakpoints set here are 4501H and 4510H. To clear the screen area prior to execution, use the C command.

C COMMAND

The C (Clear) command clears the screen. If in your program development the screen is to be used as an output device, it may be desired to have the ASCII "blank" character fill these RAM locations. Execution of the C command clears the screen. Utilization is

the same as other commands, except that no address or data fields are required. For example:

```
C<ENTER>
```

produces in the lower left corner the prompt

```
FROLIC:
```

The rest of the screen is clear. It should be noted that the sign-on with the prompt ":" is the result of return to the execution of the command mode. The CLEAR key is identical in function to the C command. Depressing this key produces a C in the command buffer. In both cases, the clear is a two-step operation that makes accidental use of this command more difficult. To view the data, the T command is used. With the T command, the prompt does not appear.

T COMMAND

The T (Transfer) command transfers the stored screen information located at the address specified by the V register (which is not a Z-80 register). For this command to be effective, the screen storage must be changed prior to using this command. This is done with the XV command as previously described. Therefore, returns from the user program caused by execution of the E, G, or Z (hardware option to be described) commands result in the information stored in the crt R/W memory being moved to the allocated area. That information may be recalled for inspection while in the command mode by using the T command. The result is scroll protection, because after a breakpoint is encountered while the microprocessor is executing object code, execution ceases, and the screen data is stored beginning at the location stored in the V register. This data is restored to the screen using the T command, just as it was when execution was interrupted. In this case, the sign-on message does not appear in the crt field. However, the cursor may be present (a function of the position of the G or E command in sequential executions).

Change the video pointer to 4C00H, and execute a G with a breakpoint. Alter the data on the screen with a series of <ENTERS>s. Now, to view the data stored in the reserved memory space (as determined by the V field at the time the G reached its breakpoint), enter,

```
T<ENTER>
```

The stored data now replaces the data on the screen. Note the change in the screen. If the executed program had cleared the screen (at this point, it does not matter how this was accomplished) the

"FROLIC:" would not appear on the crt screen. Although not apparent, the monitor is waiting to accept a new command. Any key entry will be displayed with the cursor mark reappearing after the entry. Continuation of execution using the G (or Z) command automatically returns the stored data to the screen just prior to execution.

Caution is required for the use of this command in a command sequence, for its return is always to the command entry point of the monitor. It can only be the terminal command of a command sequence, for any command that exists after it is not executed by the monitor. This is the only command that functions in this manner. Grouping of commands for execution is covered in later sections.

A COMMAND

The A (display [A]SCII) command is for American Standard Code for Information Interchange display. This means that if the character in memory has an ASCII equivalent or graphic representation (Radio Shack format), it can be displayed in a single position representative of the hexadecimal value. Even though the graphic character is not part of the ASCII code, the pattern of the graphic byte is written to the screen area. In fact, the only pattern that is not displayed is the 08H (backspace or rubout). If this code is in the field specified in the command, the byte prior to 08H is also not displayed. The crt driver program automatically removes data prior to the rubout. Since the A command is primarily intended for the display of ASCII-based text information, there is no correction for the rubout due to the limited EPROM space. The format of this command is the same as that of the D command. Two options are possible: the block format and the through format. A typical command follows:

```
A3C00 100<ENTER>
```

This command results in the display of the 100H locations of the video screen area beginning at 3C00H. These lines consist of an address followed by 32 decimal characters. The same display of ASCII characters can occur by entering the alternate format,

```
A3C00T3CFF<ENTER>
```

F COMMAND

The F (Find) command allows the user to find a single byte of data within the range specified by the address fields. Three fields are required in this format: the starting location, the ending location

or number of bytes to search, and the byte desired to be found. For example,

```
F0000T4FFFH F3<ENTER>
```

or

```
F0000 5000 F3<ENTER>
```

searchs memory for all occurrences of F3H, starting at 0000H and ending at 4FFFH. Each address at which the byte is found is displayed in the following manner:

```
0014: F3
00XX: F3
00XX: F3
```

Q COMMAND

The Q (Quest) command is similar to the F command with this important difference: a two-byte search is made. The format is the same as that of the F command, but the data is interpreted as an address, so if the following is entered into the buffer:

```
Q0040T07FF 4000<ENTER>
```

or

```
Q0040 780 4000H<ENTER>
```

the result is a search for the low-order byte 00H followed in memory by the high-order byte 40H, made over the range specified. The result should produce the following display:

```
0080: 00 40
0345: 00 40
```

This format of low-followed-by-high is a function of address data for the microprocessor. That format was preserved in the implementation of this command.

@ COMMAND

The @ (fill @) command allows the user to fill memory with a constant. Two more ways to clear the screen follow:

```
@3C00T3FFF 20<ENTER>
```

or

```
@3C00 0400 80<ENTER>
```

By necessity, this is a three-field command. In this case, the third field is the data to be deposited in memory over the range given. In

the first example, the ASCII blank (20H) is deposited. In the second example, the graphic blank (80H) is deposited. They both perform the same function. If more arguments are typed in than can be used by the command, the monitor accepts only that required, and it ignores the extra fields. The space is used as the delimiter for the last argument. This command can be used for presetting memory space to a reference byte such as 00 or FF.

You must be very careful when using the fill command. If a block-type format is used, it is all too easy to wipe out important data such as the user program, saved video information, or other critical data.

H COMMAND

The H (Hexadecimal arithmetic) command is used for hexadecimal arithmetic. The format contains two fields. The display first shows the result of the second argument added to the first, and then the second argument subtracted from the first. This command is useful for computing offsets for loading from cassette tape. An example of its use:

```
H9000 3000<ENTER>
```

results in the following display:

```
C000:6000
```

N COMMAND

The N (iNput) command is used to read the status of any input ports that may be attached to the TRS-80. The format also allows reading a range of inputs by specifying the range using the "T" as the delimiter or by specifying the number using the <SPACE> as the delimiter. To show an example of the format, port FF is used. As you know, for Model I and III systems, port FF is available with bit 7 used for data from a cassette. Also, if the system is a Model I, bit 6 is used to indicate activation of the large graphic mode. For a Model III system, port EC, bit 2 serves this same function. Unless other ports have been decoded (as is the case with a Model III system), reference to any port other than FFH will result in an FFH display, since the dataform of unused inputs is detected as all ones. An example of a single port input with a Model I is:

```
NFF 1<ENTER>
```

The display on the crt screen is

```
FF: 7F
```

or

```
FF: 3F
```

depending upon the activation state of the large-graphic feature. On a Model III system the display is

```
FF: 40
```

The N command may respond to a range of inputs by default if no range is specified. The default range is 40H, as with the D and A commands. If one of the following formats is used:

```
N00T05<ENTER>
```

or

```
N00 5<ENTER>
```

the output is as follows:

```
00:FF 01:FF 02:FF 03:FF 04:FF
```

O COMMAND

The O (Output) command is used to output data to a port specified in the format. Of course, the hardware for the port must exist in order for the command to have meaning. The Model I and III TRS-80s have output ports already available as mentioned in an earlier chapter. As an illustration of the use of the O command, the large-graphic feature is enabled through a port access. In a Model I system, a 08H written to port FF enables this feature. In a Model III, a 04H to port EC enables the graphic. The following example shows how to enable the large-graphic display in a Model I system.

```
OFF 08<ENTER>
```

or, for the Model III

```
OEC 04<ENTER>
```

The screen now shows every other location (the even locations). In a Model I system, the keyboard scan checks for large-graphic enable and compensates by inserting the additional increment in the memory pointer. This is true only for the Model I system because of the limited EPROM memory that required more space in the Model III computer. Increased visibility due to the larger size is an apparent advantage through the use of this command on the Model I. In either case, previously written data or data transferred using any move-memory type of command displays only the even locations of the screen R/W memory.

The graphic feature is reset through output of 00H to port FF for the Model I, or to port EC for Model III. Any command involving cassette operations, any command with a syntax error in its format, or any command that causes a power-on clear resets the large-graphic feature.

A second example using port FF on the Model I or port EC on the Model III activates the cassette motor. Using the O command, execute

```
OFF 04<ENTER>  (Model I)
```

or

```
OEC 02<ENTER>  (Model III)
```

The cassette motor is now active. By using the O command, the motor can be turned off by resetting the appropriate bit.

M COMMAND

The M (Move) command is used for block moves of memory from one location to another. Three fields are required to execute this command. The format for the first two fields is the same as for the A and D commands. That is, the format consists of a beginning address, a space, and a block length, or the beginning address through the ending address. The third argument is the beginning address of the destination block. Be careful not to destroy data by improper use of this command. The following example shows both ways of performing block moves:

```
M6000T601F 7000<ENTER>
```

or

```
M6000 20 7000<ENTER>
```

In this example, the 20H bytes located starting at 6000H are moved to 7000H. In many cases duplicating your program in memory saves re-entry of the object code if a system crash occurs. Of course, there are malfunctions that destroy everything in the memory space, in which case nothing is gained by making a backup copy of your program in RAM. A long program could be saved on tape as an alternative option.

V COMMAND

The V command compares and verifies one block of data against another. It is automatically called if a block transfer is used. This

instantly informs the operator if an attempt is made to move to read-only memory. The format for the V command is the same as that of the M command. Three fields are required: a start address, an end address or block length, and the location at which the verification is to begin. An example is:

```
V6000T601F 7000<ENTER>
```

or

```
V6000 20 7000<ENTER>
```

Any difference that exists between the blocks is printed on the listing device. A display occurs only if the two locations differ. For example:

```
6013: 67 7013: D3
```

W COMMAND

This command and the next command concern themselves with the writing of programs on cassette tape and the reading of those tapes back into the system. The format used in the transcription of these tapes is the same as that used by the TRS-80 Level II system. The speed is 500 b/sec. In producing a tape, a name may be assigned. Up to six ASCII identifiers are possible.

The W (Write) command is used to produce a tape in the TRS-80 format. To execute the command it is entered as follows:

```
W4400 100 4400 NAME<ENTER>
```

or,

```
W4400T44FF 4400 NAME<ENTER>
```

The use of the name is optional, but an execution address must be given, or improper operation results. The space delimiter precedes the name to indicate that a name is to be written. If not present, the space allocated for a name on the tape is filled with 20H, the ASCII representation of a blank character. The execution address does not have to be the same as the starting location. However, the execution address is placed in the program counter register when the tape is read using the R command. To terminate the W command for any reason, simply depress the <SHIFT> key.

R COMMAND

The reading of a cassette tape occurs by using the R (Read) command. This command has an address field associated with it that

is optional. If an address is entered, the data is interpreted as an offset and is added to the address that is read from the tape at the beginning of each block. This allows the tape to be loaded in any available R/W memory. It is particularly useful for the loading of data to be used for the programming of EPROMs. In these cases, the execution address is often in locations other than that available as R/W memory in the TRS-80 system. In computing offset, the H command is used.

The format of the R command is,

R<ENTER> (no offset)

or,

R4200<ENTER>.

In the latter case, 4200H is added to each address during the reading sequence. If the tape is being read properly, the name appears after proper detection of the sync marker. No name results in a displacement of the cursor by six spaces. If no error occurs during the read, the execution address appears after the name (if present), and control returns to the monitor. The program counter register is also loaded with the starting address. As a consequence, execution of the loaded program can proceed by entering a G, E, or Z (optional hardware required) command without the address in the format.

An error during the R command results in a buffer display with a "?" in the location of the R command. This error results from improper adjustment of the level control on the recorder or a bad recording. The error itself is usually an improper checksum. Also, there are cases in which the sync mark cannot be detected. Depressing the <SHIFT> at any time terminates the R command.

L, P, AND U COMMANDS

The next three commands require hardware modifications to the TRS-80 system. The L command allows hard copy, and more information on hard-copy devices is given in Chapter 3 along with software support. The P command allows EPROMs to be programmed. Hardware and software to support this feature are given in Chapter 4. The last command in this section is the U command. It provides an access to system R/W memory and return to the monitor for user defined commands. A fourth command (Z) is available to allow for a single-step function; the explanation and hardware requirements are given at the end of this chapter.

L Command

The L (List) command is used for listing on a hard-copy device. It is assumed that a hard-copy device is available and the appropriate software support is loaded into the system. The L command acts as a switch and is activated any time the L command is executed in a command string. The first time the L is encountered, it turns on the hard-copy feature. The next time the L is executed, the listing on the hard-copy device stops. This command requires no address fields. For example,

```
L<ENTER>
```

is all that is required to activate the hard-copy device. The appropriate firmware (software plus hardware) must be available to support the option. Output may be through the cassette terminal and is available as a 20-mA current loop or as RS-232C voltage levels. A possible hardware interface is shown in Chapter 3. If this option is not used, this command may function as a user specified command. The user specified program should start at 4018H. A three-byte jump may be inserted to cause execution of the driver program or the user specified program. The rules for using this jump and return from it are covered in the U command and in the software sections dealing with the printer.

P Command

The P command is used to program EPROMs. Chapter 4 deals with the hardware required to provide this feature. This command, along with the U and L commands, does not have a resident program in the monitor. However, upon power-up initialization, returns (RET a C9H) are inserted into locations which normally would provide the monitor access to a user program. If the options are to be used, a jump (C3H) to the address of the user supplied routine must replace the appropriate C9H in the R/W memory referenced by the command. The address of the jump for EPROM programming is 401BH through 401DH. Greater flexibility in the programming of EPROMs is thus provided. For example, the programs used for programming 2716 EPROMs, 2708 EPROMs, or 8755 EPROMs can be entered as presented in Chapter 4 (Chapter 4 is devoted to the PROM programmer). The addresses of execution are loaded into the referenced vector when the tape containing the EPROM programming code is loaded into memory. The reset does not clear these addresses. These locations are cleared through one of the following operations: a power-on clear when the system is turned on, or the execution of a G0000. The latter assumes that the monitor resides at 0000H.

To use this command, the user must first load the EPROM program routine. This can occur at any time prior to the execution of the command. Depressing the P followed by the <ENTER> transfers control to the EPROM programming routine. Depending upon the software used to drive the programmer, fields may also be included in the entry when this command is executed. The software in Chapter 4 does not pass information to the programmer using the address fields.

U Command

The U (User) command is identical in operation to the P command, but in this case a user written program resides at the address specified. This program must be loaded, of course, previous to its execution, and RAM locations 401EH to 4020H must contain a jump instruction (C3H) followed by the address at which the user program resides.

Another consideration that must be included in the object code is a correction in the buffer pointer that resides in the IY register. Commands without fields do not advance the buffer pointer. Although the command you insert executes (if correctly written), upon return the monitor expects a comma in the buffer. Since the buffer is not advanced, the unexpected U causes the buffer to be displayed with the "?" replacing the U. This may produce strange behavior if a BS command (to be covered in the next section) is used at the end of a command sequence.

Two methods to correct the pointer may be employed. The first uses any subprogram in the monitor that accesses the buffer and therefore increments the pointer. For example in the listings in Appendixes D and E, the programs labeled GHL, G2N, G3N, and GET all advance the buffer pointer. Remember, you are not leaving the monitor with the register states changed as happens in the G, E, and Z commands. Using the U command is part of the monitor code, and, therefore, any change to a register is transparent to you and not observable with the X command. The other method is to increment the data in the IY register by using the instruction INC IY (FD 23) in your driver program. Here is an example of a program that provides a short time delay between commands. The code is loaded starting at address 4300H.

```
4300 FD23     DELAY   INC IY          ;ADJUST POINTER
4302 210000           LD HL,0000H     ;INITIALIZE COUNT
4305 25       DEL     DEC H           ;DECREMENT HIGH
4306 C20543           JP NZ,DEL       ;DO UNTIL ZERO
4309 2D               DEC L           ;NOW DECREMENT LOW
430A C20543           JP NZ,DEL       ;DO UNTIL ZERO
430D C9               RET             ;BACK TO MONITOR
```

To use this program, load the code using the I command. At the same time change the return at 401EH to C3H and insert the 00H and 43H in the next two locations. Keep in mind the low-high format. Transfer is now possible from the monitor to the delay program. The following command sequence accomplishes both requirements:

```
I4300 FD 23 21 00 00 25 C2 04 42 2D C2 04 42 C9 401E/ C3 00
43<ENTER>
```

The U command is ready for testing. The jump to this command does not change unless a G0 or power-on clear occurs. To use the U command enter,

```
U<ENTER>
```

Notice the short delay before the sign-on appears on the monitor screen. Save this program, for it is used in the demonstration of the commands B and BS.

It was already indicated that all unused commands in the command table are available for use. The buffer pointer must be adjusted if no reference to the buffer is made. Although there is no link provided in R/W memory for these, as is the case for the L, P, and U commands, the use of these commands may be acquired by reprogramming the EPROM that contains the monitor program. Chapter 4 shows how this is possible.

BUFFER

In the description of the commands presented so far, reference has often been made to the buffer or to the buffer string. The monitor presented in this chapter is in fact an interpreter, and all of the commands described may be entered as a series of commands forming a command string. You may exercise this option by using the comma "," in place of the <ENTER>. The latter occurs only once, and that is at the end of the sequence. While entering commands in the buffer, the rubout or backarrow allows removing improper entries. If the command sequence is extremely large, the use of the <BREAK> allows exit without execution. There are 256 decimal locations reserved for the buffer, so an upper limit to the number of commands is imposed. An example demonstrating the use of the command string follows. This command sequence enters object code in memory, displays the code, clears the crt screen, relocates the saved video information for recall, executes the program using the just cleared screen, returns at the selected breakpoint, and transfers the video information back to the screen as it was upon reaching the breakpoint.

```
I4800 3E 41 2A 00 3D,D4800 5,C,XV4C00,E4800.+5,T<ENTER>
```

Data is entered directly from the command string. The execution of this short program results in the 41H or "A" being written at location 3D00H, which is in the first position of the fourth line. This does not occur until the E4800 is executed. When the <ENTER> is depressed, the string is executed as follows: First the data is entered with the I command, the data is displayed with the D command, and then the screen is cleared with the C command. The XV4C00 provides a place in RAM to save the data on the screen upon reaching the breakpoint. The actual execution of the program is accomplished with the E4800. With the T command the result is viewed: a blank screen and the "A" as described. This termination of the command string with the "T" is the only valid position of the T command. It can be used alone for transfer after a command string has been executed.

It was indicated earlier that you could enter data interactively from the keyboard through the use of S4800 (in lieu of I4800) and by omitting the data sequence. However, keep the rest of the entry as shown in the example. The data for executing the short program may now be entered through the keyboard with the <ENTER> now used to deposit the data in memory. Upon completion, depress <ENTER> twice. This causes execution of the remaining string as in the first example.

If the delay program (given in the U command section) is resident in the monitor, this second example may be more convincing as an illustration. In the command sequence, the delay program is considered the program developed, and it is tested as follows:

```
X,G4200.+2,U,G.+3,U,G.4206,U,G.4209,U,G.420A,U,G.420D<ENTER>
```

The delay is short, so pay attention to the H and L registers. Breakpoints are placed after critical instructions. This demonstration sequence illustrates the use of the monitor as a development tool. The X command at the start of the sequence displays the status of all the registers at the start of execution, and therefore, when G4200 is executed, the data in these registers is transferred into the Z-80 registers. At the first break, after the INC IY instruction, the automatic display of the internal registers shows that the only register altered is the Y register, and its value has been incremented by one. The next breakpoint shows the H and L registers initialized to 0000H. The following breakpoint occurs after the H register is decremented from 00H to FFH. This is the first instruction to alter the flag register, and the Z flag is absent from the flag display. (In the Z-80 instruction set, double register increments and decrements do not affect the status of the flags.) The break at 4209H occurs after the H register is dec-

remented 256 times, and the zero flag is observable. The last break is after 65,536 decrements. The H and L registers are now both zero, and the zero flag is zero. The last command in the sequence is executed, and control is passed back to the monitor for the next command instruction.

A third example of the use of the command sequence further demonstrates its power. In this example, we demonstrate the usefulness of the monitor as a teaching device to aid in understanding the instruction set of the Z-80 processor. Assume for the moment you wish to identify the difference between the compare and subtract instructions. The data in the accumulator is less than the data it is to be compared with, which resides in the B register. Enter the following command sequence:

```
I4300 B8 90,XF00,XA85,XB9E,H85 9E,G4300.+1,XA85,XF00,
G.+1<ENTER>
```

The object codes for the two instructions (compare A register with B register [B8H] and subtract B register from A register [90H]) are entered at 4300H and 4301H. The flag register is cleared of flags (XF00), and the A and B registers are initialized. The H command is used to see the sum and difference of 85H and 9EH, and the compare is executed. Since A is less than B, the carry is set. This is a convention opposite to what you would expect in twos (or tens) complement arithmetic, but more familiar to the process you use in decimal subtraction. The compare is accomplished with a subtract. This operation does not use the status of the carry (borrow), similar to other logic instructions. Furthermore, the accumulator is not altered. Since the two numbers are different, the zero flag is not set. Also the second number is greater than that in the accumulator. Therefore, a carry (borrow) is generated. In subtract operations, the meaning of the carry flag is the same as a borrow. Since this operation was a subtract, the N flag is set. The other flag set is the H flag, and it would have meaning if the original entries were binary coded decimal. The overflow flag is not set, since the number is representative of the subtraction within the range of −128 to +127. In this case a −98 (9EH in twos complement) was subtracted from a −123 (85H), leaving a −25 (E7H). The subtract instruction that follows shows this to be true. It was not necessary to initialize the accumulator with the 85H since this data was not destroyed by the compare instruction. After the subtract is executed, you will notice that the accumulator now contains E7H (the H command shows the difference to be FFE7H), and the flags are the same as for the previous compare.

The addition of the command sequence buffer to the monitor is

its most powerful attribute, and, as a result, two additional commands are available.

BS AND B COMMANDS

The BS (Buffer Save) command saves the command sequence in a reserved area for repeated execution. You simply enter a BS as part of the string and the buffer is saved during execution of the sequence. To execute the saved buffer, a B (execute Buffer) command, transfers control to the buffer, and execution of the buffer begins immediately. The B should terminate the sequence, since any instruction that follows the B cannot be executed. The reason for this is that after the transfer execution begins with the first command in the buffer. It is quite easy for you to enter into endless loops with this command. For instance, the following loops upon itself, saving itself and executing itself endlessly.

```
BS,B<ENTER>
```

To regain control of the command buffer, depress the reset switch. The use for this command is unlimited. Although there can only be one saved buffer sequence resident in the monitor, nothing prevents the user from saving this sequence with the move command (M), or storing the buffer on cassette tape. To aid you, you should know that the buffer is located from 4180H to 427FH. The memory after these addresses, up to 4300H, is used for stack storage. One can use the move command to change the saved buffer to regain a desirable sequence of commands, then execute the B command. Testing hardware involving input, output, or a combination is possible with the command. For example, to check for proper address decoding in your interface design, you could execute the sequence:

```
BS,N80,O2B,B<ENTER>
```

This results in the "strobing" of the device address signals for ports 80H and 28H along with IN/ and OUT/ signals. To exit from this loop, depress the <BREAK> key.

A second example of this command shows entry of complex breakpoints. The program counter must be initialized with the XP, G, or E commands. Entering the sequence

```
BS,G.4303.4304.4309.43E2,B<ENTER>
```

results in register displays in the order the breakpoints were reached. Holding of the space bar allows examination of the data in the registers.

Continuous single-stepping in read-only memory in addition to R/W memory is possible if the single-step hardware has been imple-

mented. The description of this next command completes the chapter.

Z COMMAND (SINGLE-STEP)

The Z (single-step) command requires a hardware modification to implement. It is a most attractive addition to the command set, for this is one way to execute instructions in ROM. Coupled with the BS and B commands, the Z command aids significantly in software development. The execution address, which loads the program counter, can be initialized with the same convention used in the G or E commands. If the hardware is not present, a breakpoint should be included, and the command will function. However, one obvious and significant difference occurs if you do this: the program counter decrements upon return, because the instruction at the breakpoint address is never executed when using the breakpoint feature. Instead, the single byte call of the RESTART instruction replaces the normal instruction residing at that address. However, the Z command, or single-step, in contrast, executes the instruction located at the position of the current program counter. The instruction may be any length and any type. Jumps or calls, single or two byte loads, and even those four byte Z-80 codes are all executed. The instructions may exist in ROM or in RAM. The format for this command is

```
Z<ENTER>
```

or

```
Z4800<ENTER>
```

However, there is one feature you must be aware of when using the single-step option. If the instruction to be executed is a disable interrupt, return to the monitor will not occur since the hardware uses the interrupt to accomplish this end. (If the execution is in R/W memory, a breakpoint may be appended to the Z command, and return will occur at this breakpoint. If this does occur, the program counter as displayed by the P register will be one greater than the breakpoint. An adjustment must be made to correct this register to continue with proper execution.)

The mode of interrupt should remain mode 1. The other two modes, 0 and 2, may work. In mode 0, the interrupting device is expected to place any instruction on the data lines during the INTAK/ time, and the microprocessor executes it. By default, in the Model I an FFH is strobed on the data lines, unless you have altered the system with additional hardware to strobe these data lines. An FFH instruction is an RST 38H, the same as used for the breakpoints. Thus,

the return back to the monitor will be to the register save routines. This is not the case with a Model III; if the mode of interrupt was changed to mode 0 under a user software routine, then hardware must be supplied to accomplish the strobing of an FFH onto the data lines. In interrupt mode 1, the data lines need not be strobed, and the microprocessor responds automatically to the interrupt by executing a restart to location 0038H. This is the mode of interrupt initialized by the monitor during power-up.

You can use mode 2, but the LSB of an address and not a restart is strobed onto the data lines during the INTAK/ time. The MSB of the address is obtained from the (I) vector. (Refer to the TRS-80 technical manual for more detail.) This address must point to a location that will cause execution to take place at the register save routines in the monitor program. Hardware must be supplied, and in addition if the unit is a Model I, the interrupt alteration must be made. (Refer to Chapter 1.) This last method for single-stepping should be avoided unless absolutely necessary for your program development, at which point you must design the hardware to support the use of this mode of interrupt.

An example of the usefulness of the Z command is shown by using the following sequence:

```
BS,Z,B<ENTER>
```

The program counter should be initialized to the starting address external to this string, and the stackpointer should not conflict with the monitor stack. The crt will display the register status as each instruction is executed. If a hard-copy device is available and the driver program loaded, the addition of the L command to the sequence will provide a hard-copy output.

```
BS,L,Z,L,B<ENTER>
```

The L could also be activated external to the command. The listing device will trace the execution of each step in the program until you cease its operation using the <BREAK> key.

The hardware modification for the single-step addition is shown in Fig. 2-1. These modifications can be made internally by altering the CPU socket or externally by using the system bus signals available at the edge-card connector. Two design possibilities exist, as shown in Fig. 2-1. The single-step hardware must use a port to activate the shift register. The port chosen for this application is port 00H. A slight difference exists in the designs between the two TRS-80 models. For the Model I, the instruction fetch cycle signal (M1/) is not available at the edge-card connector. Instead of using the fetch cycle to advance the low logic level through the shift register, the

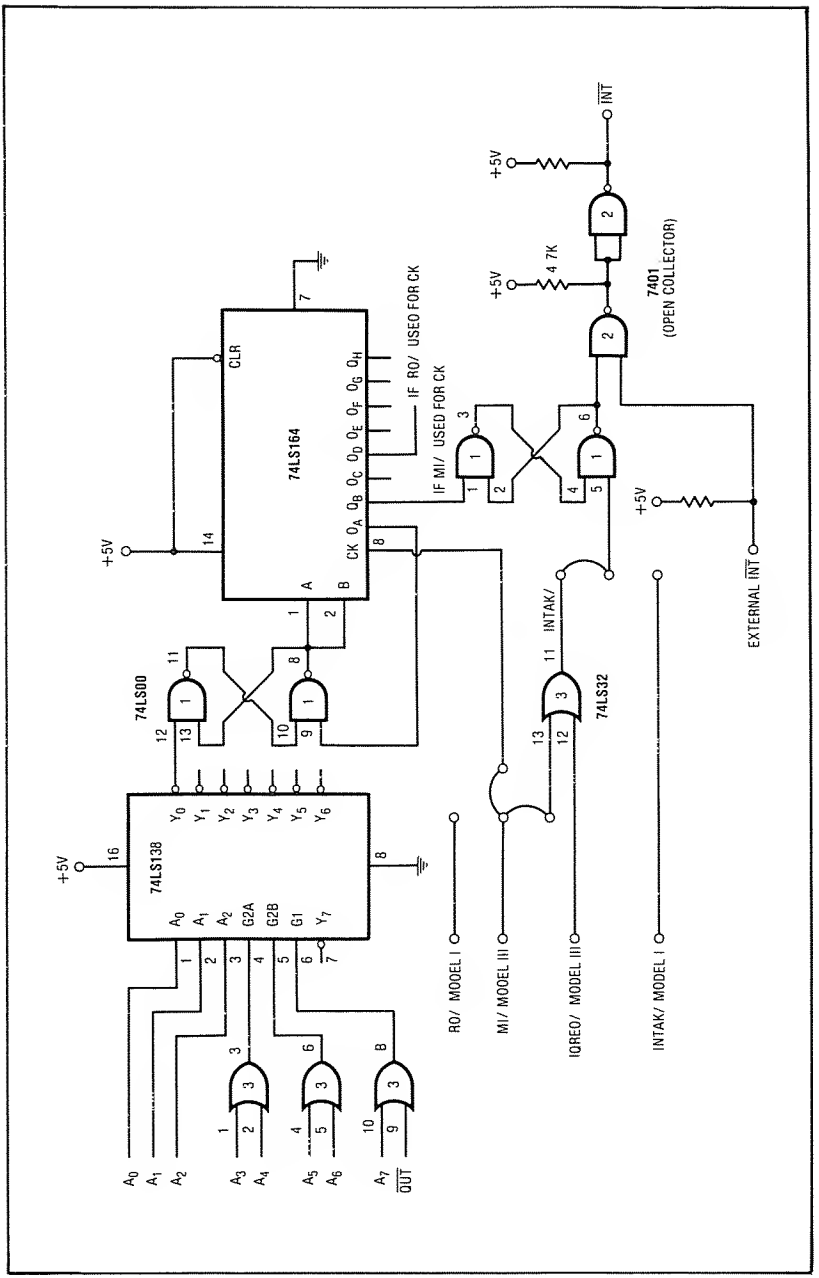


Fig. 2-1. Single-step circuit diagram for TRS-80 Model I and Model III.

RD/ signal is used for this function. Because there are more memory references, a different output from the shift register is required. The tapped positions for these control signals are referenced in the schematic. The complete design is shown in Fig. 2-I.

The theory of operation of this circuit is to use a port write to activate a low level in a data latch, then shift this low through a shift register. The data latch is cleared after the first shift using output Q_A of the 74LSI64. If single-step is required, the port write is to port 00H (the only time this port is accessed by the monitor). The execution of the Z command sets a flag, and if true when tested the EI and OUT (00H) A instructions are executed. If the flag is not true, these instructions are bypassed. For single-step to occur, the low bit from the latch must be shifted through the shift register. For each MI/ or RD/, this low bit is shifted. You simply count the number of memory references (MI/ or RD/) and tap the shift register at that point in order to produce the single-step interrupt. A second data latch is used to hold the interrupt request (INT/) line low until it is acknowledged by the CPU (INTAK/), which then resets the data latch and returns the INT/ to the high logic state. In the monitor as written, there is one instruction with three memory references before the single-step interrupt can occur. Depending upon the TRS-80 model used, the instruction fetch (M1/ for the Model III), or the memory reference (RD/ for the Model I) produces an interrupt at the appropriate time resulting in the execution of a single instruction. Of course, if there is a possibility of interrupts from other sources that may have been implemented on the system, a break may occur somewhere between the JP (JUMP) instruction and the last memory reference. If this is a possibility, examine the program counter upon return. This register is your indication of what has occurred.

Key elements in the hardware design are the port decode and shift activization. The I of 8 decode (74LS138) was chosen to give flexibility in port choice, but modifications in the software must reflect any port change. The shift register is constantly shifting a high through it each time an instruction fetch or memory reference is made. The 74LSI64 shifts on the positive edge of either of these two signals. The moment execution reaches the user's program, a latch is set to activate the interrupt line. This latch is cleared with the interrupt acknowledge signal. The interrupt signal from the shift register has provision to be gated with an external interrupt to allow for greater flexibility in system design.

CHAPTER 3

Hard Copy From the Monitor

When editing programs, it is often useful to have a listing of the instruction codes on paper. One of the cheaper sources for hard copy is the teletypewriter. Though slow, it has a proven record of reliability. A software option is included in this chapter: a teletypewriter driver routine. To use this software option, a hardware interface is required. A circuit is provided that allows two-way communication with a serial device through the cassette DIN plug. If only the serial printing option is required, then the hardware that converts the teletypewriter data transmission signals to TTL levels (suitable for input to the TRS-80 using the cassette plug) can be omitted. To activate the teletypewriter list option, a program must be entered into memory. The program and the loading steps are given later in this chapter. The L command is used to activate the printing device, as described in Chapter 2.

The program given drives a serial device, which may be compatible with either RS-232C voltage levels or 20-milliampere current loops. Implementing the circuit is straightforward, since the control of the circuit is determined by the software.

The hardware schematics for serial input/output are shown in Figs. 3-1, 3-2, and 3-3. For serial transmission to a printer, a comparator converts the audio cassette signals to bipolar voltage levels that are compatible with the RS-232C interfaces in most serial printers. Since a few printers require a 20-milliampere current drive, additional hardware is used to generate the necessary current levels from the bipolar voltage signals. The hardware that receives the

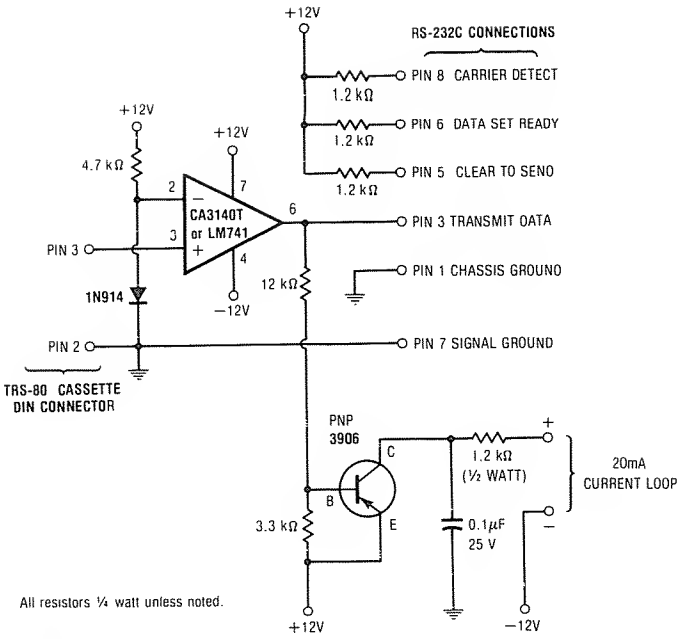


Fig. 3-1. Printer driver circuit for RS-232C and 20-mA current loop.

serial data from the teletypewriter, as shown in Fig. 3-3, is more complex. Because of the nature of the TRS-80 cassette input and the format of serial data, an unconventional circuit is required. The details describing this circuit are presented later in the chapter.

The most universally accepted format was established by the Electronic Industries Association (EIA). The voltage format is known as the RS-232C, the C indicating that this is the latest revision. The Bell System developed this standard in cooperation with the EIA as the standard for interface between data terminal equipment and data communication equipment employing serial binary interchanges of information; that is, interchanges of one bit at a time.

According to the standard, the voltage level under open-circuit conditions must not exceed a magnitude of 25 volts. The driver asserts a voltage that is between −5 V dc and −15 V dc relative to signal ground to indicate a MARK condition (logic level 1), indicating the presence of a data bit. The driver asserts a signal level that is between +5 V dc and +15 V dc relative to signal ground to indicate a SPACE condition (logic level 0), indicating the absence of a data bit. Often chosen for the dc levels of MARK and SPACE are −12 V dc and +12 V dc. There are loading requirements in addition

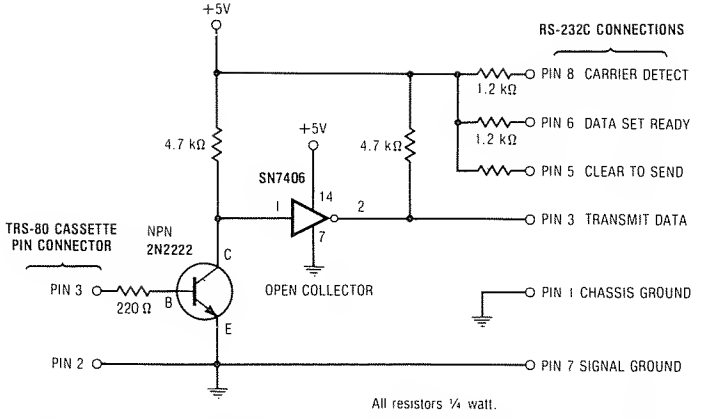


Fig. 3-2. Alternate printer driver for RS-232C (not to specifications).

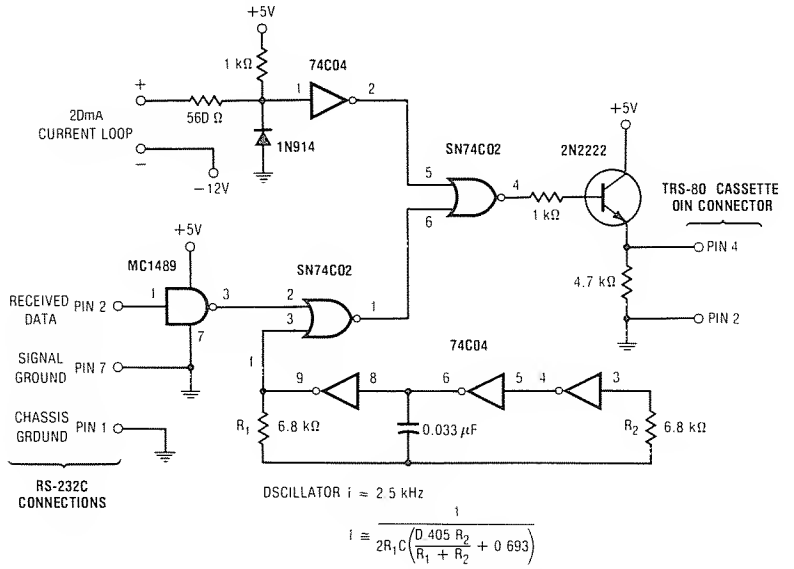


Fig. 3-3. Input circuit for the reception of RS-232C and 20-mA current loop signals by the TRS-80.

to the requirements for specific voltage levels. The load the receiver presents to the driver should be between 3000 and 7000 ohms. The driver must be able to function under the load presented by the receiver and maintain voltages within the dc levels of +15 V dc and −15 V dc. Under no load conditions may this level rise above the 25-volt limit.

Physically, a 25-pin plug is often associated with this standard. The signal assignments for these pins are shown in Chart 3-1. Pins 2 and 3 change their orientation depending on whether the device is the originating equipment or the terminal equipment.

Chart 3-1. RS-232C Interface Circuit Functions

Pin 1. Protective Ground — Electrical equipment frame and ac power ground.
Pin 2. Transmitted Data—Data originated by the terminal to be transmitted via the sending modem.
Pin 3. Received Data—Data from the receiving modem in response to analog signals transmitted from the sending modem.
Pin 4. Request to Send (RTS)—Indicates to the sending modem that the terminal is ready to transmit data.
Pin. 5. Clear to Send (CTS)—Indicates to the terminal that its modem is ready to transmit data.
Pin 6. Data Set Ready (DSR)—Indicates to terminal that its modem is not in a test mode and modem power is on.
Pin 7. Signal Ground — Establishes common reference between modem and terminal.
Pin 8. Received Line Signal Detector (LSD)—Indicates to the terminal that its modem is receiving carrier signals from the sending modem.
Pin 9. Reserved for test.
Pin 10. Reserved for test.
Pin 11. Unassigned.
Pin 12. Secondary Received Line Signal Detector—Indicates to the terminal that its modem is receiving secondary carrier signals from the sending modem.
Pin 13. Secondary Clear to Send—Indicates to the terminal that its modem is ready to transmit signals via the secondary channel.
Pin 14. Secondary Transmitted Data — Data from the terminal to be transmitted by the sending modem's channel.
Pin 15. Transmitter Signal Element Timing —Signal from the modem to the transmitting terminal to provide signal-element timing information.
Pin 16. Secondary Received Data—Data from the modem's secondary channel in response to analog signals transmitted from the sending modem.
Pin 17. Receiver Signal Element Timing—Signal to the receiving terminal to provide signal-element timing information.
Pin 18. Unassigned.
Pin 19. Secondary Request to Send—Indicates to the modem that the sending terminal is ready to transmit data via the secondary channel.
Pin 20. Data Terminal Ready (DTR)—Indicates to the modem that the associated terminal is ready to receive and transmit data.
Pin 21. Signal Quality Detector — Signal from the modem telling whether a defined error rate in the received data has been exceeded.
Pin 22. Ring Indicator (RI) — Signal from the modem indicating that a ringing signal is being received over the line.
Pin 23. Data Signal Rate Selector—Selects one of two signaling rates in modems having two rates.
Pin 24. Transmit Signal Element Timing—Transmit clock provided by the terminal.
Pin 25. Unassigned.

The current loop is another widely used standard. It originated with Jean-Maurice-Emile Baudot, a French engineer. The principle in this communication link is the maintenance of a constant current. Two conditions are possible; the current flows, or it does not. Continuity, or current flowing, denotes the MARK. The interruption of flow denotes the SPACE. At first, when the link is established, cur-

rent flows. An interruption of the circuit indicates the start of data transmission. The current on the line is switched on and off, and in this manner it is possible to send code for the character that is transmitted. A series of stop bits or logic ones indicates to the receiver that the code has been transmitted. The line remains closed until the start of the next character transmission. This method of sending information in single characters, rather than in blocks of characters, is called *asynchronous communication.*

A popular current level used to indicate the presence of a data link is 20 mA, and you will often hear the reference made to the 20 mA current loop communication link.

The communication between the monitor and the listing device is asynchronous. By convention, serial teleprinter systems have adopted the idle, or wait, condition to be indicated by the MARK. Synchronization for a word of transmitted data is indicated by the presence of a start bit or SPACE, followed by 8 bits of data. The bit sequence for each character ends with a MARK which is transmitted for at least one bit time, and often for two, depending upon the convention that has been adopted for the communication loop. In true asynchronous form, the line remains in the MARK state until the next data word is to be transmitted. This format is the same for both standards, either voltage or current. The bit times are determined by the rate at which transmission is to occur. A typical sequence of logic levels for the 11-bit serial data (start bit, 8 data bits, and 2 stop bits) is shown in Fig. 3-4. Data rates are often specified in baud, in honor of Jean-Maurice-Emile Baudot, who originated the Baudot 5-bit code.

The characteristic that is important in design considerations for the TRS-80 is the constant level of current or voltage if no bit changes occur in the serial transmission. For the TRS-80 to receive

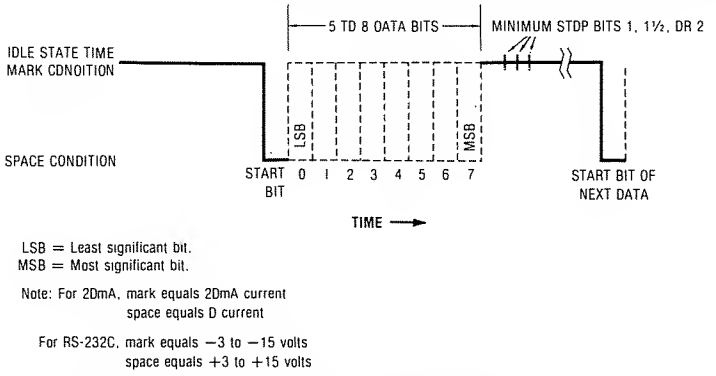


Fig. 3-4. Serial data format for RS-232C and 20-mA signals.

this code, it is necessary to introduce a level change. This is covered in the section dealing with the hardware.

The software supporting the driver for the monitor assumes a data rate of 110 b/sec, which is the traditional speed used for teletypewriter-based communication links. This data rate results in the transmission of ten characters per second. A character is defined by an 8-bit code. The code is not Baudot, but the newer American Standard Code for Information Interchange (ASCII). This is a 7-bit code, and the additional bit transmitted is used for simple error detection. This additional bit is called the *parity bit*. Its value, a one or zero, is used to make the sum of MARK bits in the word equal to an even number (in which case even parity exists), or to make the number of MARKs in the word an odd number (to indicate odd parity). For example, the 8-bit ASCII value for the NUL is 00H if parity is even, and 80H if parity is odd. By convention for teletypewriter systems, there is one start bit and there are two stop bits, in addition to the eight bits required for the ASCII character. This makes the number of bits transmitted per character equal to eleven. The bit time for transmission is 9.09 ms.

The hardware shown in Fig. 3-1 converts the output from the cassette DIN plug to the EIA levels. In Chapter 1, it was shown that the output to the cassette is controlled by the logic levels on bits 0 and 1 of port FFH to produce a voltage on pin 3 relative to pin 1 on the DIN plug of the TRS-80. To refresh your memory, the following three states are used. A 00 (00H) in the last two bit positions produces a dc level of 0.45 V, a 01 (01H) produces a 0.85 V level, and a 10 (02H) produces a 0.0 V level. This voltage is applied to the noninverting input of an operational amplifier (CA3140T). The amplifier is operating at very high gain, and in this design functions as a comparator. The reference voltage on the inverting input is obtained from a forward-biased signal diode (1N914). The reference is approximately 0.6 V. The normal output to bits 0 and 1 of port FFH is 00, and the 0.45 V produced is not sufficient to saturate the amplifier to +12 V. Therefore, the output to the RS-232C connector is −12 V, the MARK level. Only the binary 01 in the last two bits produces the "true" SPACE (+12 V level) and triggers the start-of-transmission signal. If a standard RS-232C connector is used on the printing device, the appropriate levels are applied to the Clear to Send, Data Set Ready, and Carrier Detect lines to allow the listing device to respond to a transmitted signal. The hardware also provides the transistor circuit needed for a 20-mA current loop. The voltage or current levels are a function of the hardware, but the software that transmits the serial data to the listing device is the same, no matter which transmission standard is used.

Fig. 3-2 shows an alternate form of the RS-232C interface. This does not meet the RS-232C standards, but the circuit functions well for most serial interfaces. Its most outstanding feature is its simplicity. Any voltage level less than +3 V dc results in a negative assertion, or MARK condition, on the RS-232C line.

A software listing to produce hard copy is given. The software is activated by a toggle-type switch. This switch sets a flag, and when the processor is executing code at the end of the crt routine, it tests the flag to see if a hard-copy listing has been requested. If so, a call to the hard-copy routine is executed, and a single character is sent to the listing device. The toggle switch is activated by the L command in the buffer during command execution. The hard-copy flag is set and reset with the L command. Of course, the L may stand alone or appear several times, but each time it is encountered it turns on or off the listing device. It is necessary for you to patch a three-byte jump instruction in the referenced R/W memory vector location to cause transfer to the program that supports your output device. This jump must be placed in locations 4018H through 401AH. The actual driver software may be placed in the RAM area or in ROM. Locations from 0800H through 37DFH in a Model I system are available if the monitor resides in the first 2K of ROM. In a Model III system, locations up to 37FFH are available, but you must avoid location 37E8H. This location in memory is used by the screen printer. If a Radio Shack printer is connected to your system, the status of the printer is available if you read data from this address. Data is latched in a special output buffer to drive the printer if you write data to this memory location. In the Model III, the hardware decode is supplied, so you could write your own software driver to take advantage of its presence.

SERIAL INTERFACE DRIVER CODE FOR RS-232C

An example of a driver program for the hardware described is provided. The structure of the software program is straightforward. The first output is a 9.09 ms start pulse, followed by the 8-bit data stream (least-significant bit through most-significant bit), and ending with two stop bits. This rate is the standard 110 b/sec rate. To change data rates it is only necessary to change the length of the time delay used to output the MARK and SPACE levels. Table 3-1 shows the factors to be used for delays for other data rates.

The source code shown in Example 3-1 (see page 93) provides a routine to write to the RS-232C serial device. Data to be output must be placed in the C register before this program is used. This program must be entered into RAM, because it is not located in the

Table 3-1. Data Rate Factors for RS-232C Serial Input

Baud	Model I	Model III
110	026BH	02BDH
300	00DFH	0110H
600	006CH	007BH
1200	0033H	003BH

FROLIC ROM. It has been written to reside at locations 4300H through 4339H, but it may be relocated if desired.

The program listed may be entered into the RAM locations given using either the I command or S command. The jump (C3H) is entered into location 40I8H, and 4019H and 40IAH contain the data 00H and 43H using the Intel format. The object code may be stored on cassette tape for future use with the following command sequence:

```
W4018 3 0 LIST1,W4300T4348 0000 LIST2<ENTER>
```

To load this program when hard copy is desired, the following command sequence is used:

```
R,R<ENTER>
```

You may now execute the L command and obtain a listing of all monitor activities.

SERIAL INPUT

It may also be desirable to be able to read data from a punched tape, a modem (often used for telephone links), or an external keyboard. The hardware presented in Fig. 3-3 can be used to convert the RS-232C levels from the external driver to levels that drive the input circuits of the TRS-80. This circuit may appear to be unusual unless you understand how the signals from the cassette are converted to TTL levels. The signal conditioner in the TRS-80 does not allow direct current amplification. Only ac voltages can be passed by the circuit to set the data latch. The TRS-80 circuit is a high-pass active filter with a two kilohertz breakpoint frequency. In order for the II0 b/sec signal to be transmitted to the data latch, it is gated with a higher frequency signal. The circuit consists of an oscillator and a logic gate. When the clock frequency generated by the oscillator is gated with the serial transmission, the newly formed signal is suitably matched to the electronics of the TRS-80, and this gated signal carries the data information contained in the serial signal. Fig. 3-5 shows the gated clock signal produced by the hardware. The

EXAMPLE 3-1.

```
                ORG 4300H
4300 F5         TTYP    PUSH AF         ;SAVE REGISTERS
4301 C5                 PUSH 8C         ;ON STACK
4302 E5                 PUSH HL
4303 79                 LD A,C          ;DATA IN C REGISTER
4304 F5                 PUSH AF         ;NEED LATER
4305 CD2143             CALL TPOUT      ;OUT DATA ON SERIAL DEVICE
430B F1                 POP AF          ;RETRIEVE DATA
4309 FE0D               CP 0DH          ;CARRIAGE RETURN
430B CC1243             CALL Z.LNFD     ;NEED A LINE FEED
430E E1                 POP HL          ;RESTORE REGISTERS
430F C1                 POP BC
4310 F1                 POP AF
4311 C9                 RET
4312 3E0A       LNFD    LD, A,0AH
4314 CD2143             CALL TPOUT
4317 0E04               LD C,4H         ;DO NOT NEED C ANYMORE
4319 AF         NULLS   XOR A
431A CD2143             CALL TPOUT
431D 0D                 DEC C
431E 20F9               JR NZ.NULLS
4320 C9                 RET
4321 B7         TPOUT   OR A            ;CLEAR CARRY FOR START 8IT
4322 O609               LD B,9D         ;START BIT PLUS 8 DATA
4324 F5         TP1     PUSH AF         ;SAVE
4325 D43643             CALL NC,SPACE
4328 DC3C43             CALL C,MARK
432B F1                 POP AF
432C 1F                 RRA
432D 10F5               DJNZ TP1
432F CD3C43             CALL MARK       ;TWO STOP BITS
4332 CD3C43             CALL MARK
4335 C9                 RET
4336 3E01       SPACE   LD A,01H
433B D3FF               OUT(0FFH),A
433A 1804               JR DELAY
433C 3E02       MARK    LD A,02H
433E D3FF               OUT(0FFH),A
4340 1800               JR DELAY        ;KEEP STATE TIME SAME
4342 216802     DELAY   LD HL,0268H     ;SEE TABLE FOR BAUD RATES)
4345 2B         DEL1    DEC HL
4346 7C                 LD A,H
4347 85                 OR L
4348 20FB               JR NZ,DEL1
4349 C9                 RET
```

MARK produces a series of 0.25-ms pulses, while the SPACE inhibits any pulses from passing through the gate. Thus, it is possible to latch the bits in the serial word, and write a program that can be used to convert the latched information into 8-bit words. The internal TRS-80 set/reset register used for the cassette input indicates

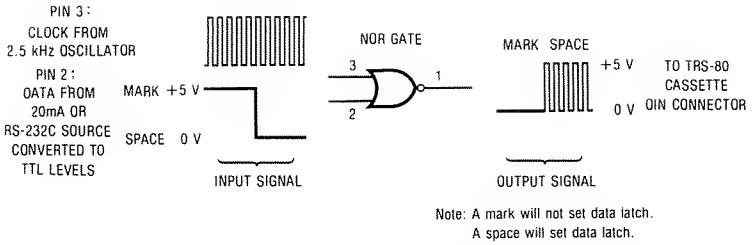


Fig. 3-5. Forming a data latch signal for the TRS-80.

the presence of a data pulse in the data stream. Before any data can be read, the latch must be cleared. After a pulse is detected in the input stream of serial bits, the latch is set. To reset the latch after each bit has been detected, it is necessary to do an output to port FFH. Port FFH, you may recall, is used for both reading and writing of data during cassette operations. In this application, we are using these inputs to detect serial data. There are upper limitations on the baud rate, but 110 and 300 b/sec are possible with the hardware shown. The software program given in Example 3-2 accepts data from the cassette input and writes it to the screen. You must alter this software to tailor it to your own requirements. If a hard-copy driver is resident, you may "echo" or transmit the data to this device. The patch to the external keyboard is made through the U command, so the appropriate jump must be placed at 401BH through 401DH.

SERIAL INTERFACE RECEIVER FOR RS-232C

Note: this program allows reading of an RS-232C signal from an external device connected to the cassette input DIN connection. Upon return from this routine, the accumulator contains the 8 bits of the received character. Because of the many general applications of this program, you may wish to modify it.

The hardware described in this chapter is not required for establishing data links between two TRS-80 computers. Data transmission using the cassette plug is possible if phase or frequency coding is used to format the data. This is the type of format used to produce a TRS-80 cassette tape. It is necessary to provide a voltage gain for the cassette output signal to raise its level to drive the cassette input of another TRS-80 computer. The input impedance to the amplifier is 100 ohms, which is relatively low, so some additional drive capabilities are required. A comparator is used to raise the level of 0.85 V dc to a TTL level. Fig. 3-6 shows a suitable driver.

We have used the TRS-80 cassette input/output for data transmission rates of 5000 b/sec. This is only possible if the TRS-80 cas-

EXAMPLE 3-2.

```
                ORG 4400H               ;START OF PROGRAM
                HALFD   EQU  0133H      ;HALF DELAY TIME
                                        ;(070H FOR 300 BAUD)
                DEL1    EQU  01CDH      ;3/4 DELAY TIME
                                        ;(0A9H FOR 300 BAUD)
                DEL3    EQU  0099H      ;1/4 DELAY TIME
                                        ;(038H FOR 300 BAUD)
                ;
4400 D9         KEY     EXX             ;SAVE REGISTERS
4401 AF                 XOR A           ;CLEAR ACC TO USE IN OUTPUT
4402 D3FF               OUT (0FFH),A    ;RESET DATA FLAG
4404 3A8038     KEY1    LD A,(3880H),A  ;SEE IF SHIFT PRESSED
4407 B7                 OR A            ;ESCAPE FROM READ
440B CA6600             JP Z,0066H      ;RETURN TO MONITOR
440B DBFF               IN A,(0FFH)     ;LOOK FOR START BIT
440D 17                 RLA             ;BIT 7 SHIFTS TO CARRY
440E 30F4               JR NC,KEY1
4410 0608               LD B,08D        ;BYTE OF DATA
4412 113301             LD DE, HALFD    ;HALF DELAY TIME
4415 CD3244             CALL DELAY      ;CENTER POSITION
4418 11CD01     BYTE    LD DE,DEL1      ;DELAY 3/4 BIT TIME
441B CD3244             CALL DELAY
441E AF                 XOR A
441F D3FF               OUT (0FFH),A    ;CLEAR DATA FLAG
4421 119900             LD DE,DEL3      ;QUARTER OF DELAY TIME
4424 CD3244             CALL DELAY
4427 DBFF               IN A,(0FFH)     ;GET DATA
4429 17                 RLA             ;PUT IN CARRY
442A CB19               RR C            ;SHIFT INTO C
442C 10EA               DJNZ BYTE       ;GET 8 BITS
442E 79                 LD A,C          ;DATA INTO AC FOR RETURN
442F 2F                 CPL             ;COMPLEMENT DATA
4430 D9                 EXX             ;RESTORE REGISTERS
4431 C9                 RET             ;RETURN TO CALLING PROGRAM
4432 1B         DELAY   DEC DE
4433 7B                 LD A,E
4434 B2                 OR D
4435 20FB               JR NZ,DELAY
4437 C9                 RET             ;RETURN TO CALLING PROGRAM
                        END
```

sette hardware patch has not been installed on the units. In addition, different data formats and lower input levels are required (0.85 to 1.2 V dc). To see if your unit is modified, examine the serial number. If it has a -01 appended, the modification has been installed and must be removed to obtain higher baud rates. In the high-speed communication link, the cassette is not used. Only for direct communication between TRS-80s and other TTL compatible inputs is the link used. For example, we use this method to down-load programs assembled on the TRS-80 to Intel SBC 80/10 and 80/20 single-board computers.

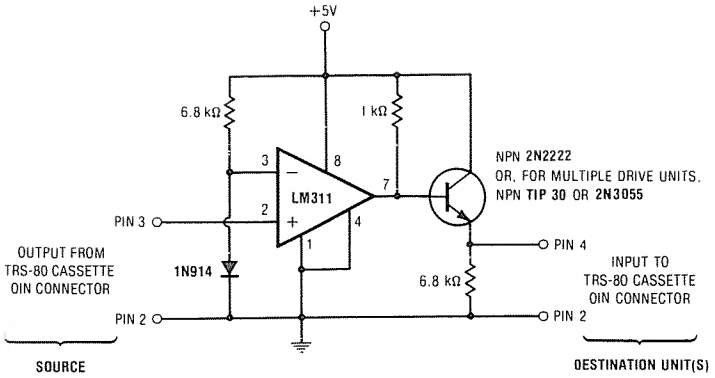


Fig. 3-6. A master-slave connection between TRS-80 computers.

Most small hobby computers use low-cost cassette recorders and cheap cassettes. In the TRS-80 Model I system, reliability is a problem. Overall system performance depends on the quality of the recorder and the quality of the tape, and not on the format used. Significant improvement in the cassette electronics allowing increased speed is provided on the Model III system. To take full advantage of the higher speed, the time delay routines must be altered in the monitor. However, to maintain compatibility between the two systems, the 500 b/sec data rate was maintained in the monitor listing.

CHAPTER 4

PROM Programmer

After an often used program has been developed and tested, you might want to store the object code in nonvolatile memory, that is, in Programmable Read-Only Memory (PROM). The use of PROM allows the system designer to remove power knowing that once power is restored the program is available immediately to perform its intended function. A software and hardware option controlled by the monitor supports the programming of PROMs, in particular both the 2708 and 2716 EPROMs. The EPROM is an Erasable Programmable Read-Only Memory. The satisfaction received in seeing the object code of your program execute in a stand-alone turnkey operation is worth the development effort. The ROM that contains the object code that defines the TRS-80 development system is one such example. Since all mnemonic listings and commented source codes are listed in the appendixes, you could now delete or add instructions to customize the system for your own personal use. Space has been provided in the command look-up table for user-generated commands. The U command as previously explained has a jump to a RAM address at which you insert a jump to your program. The P command is reserved for PROM programming. The object codes for programming several popular EPROMs are given in this chapter. If EPROM programming is not desired, this monitor command may be used for an alternative purpose. The other commands, not used by the monitor but in the command table, nevertheless, have been assigned restart instructions in place of any two-byte address. The RST 38 (FFH) is used for this purpose because an FFH state is the condition of an erased EPROM location. Furthermore, this pattern as an instruction is used for returns from breakpoints. Accidental

use of one of these instructions returns control to the monitor without loss of system control. The FFH allows you to program jumps directly into the command table without a branch to RAM location. As will be shown later, the monitor ROM is first transferred to a clean EPROM using the program option. The jumps are then programmed "over" the FFH instructions. There is provision on the TRS-80 board for an additional 2K EPROM (2716).

ERASABLE PROGRAMMABLE ROMS

The most popular programmable read-only memory to program is the erasable type (EPROM). This EPROM family was first introduced by the Intel Corporation. The availability of EPROM devices made the rapid development of microprocessor control systems possible. The only nonvolatile memories prior to EPROM were core memory, fusible-link memory, and masked read-only memory. Core memory is expensive, since it must be strung by hand. Masked and fusible devices, once programmed, are totally useless if an error is accidentally programmed into the device. With EPROM, however, high-intensity short-wave ultraviolet light can be used to erase the device. These light sources are readily available for many suppliers. Programming errors are more frequent than one would care to admit.

EPROM Physics

An elementary description of how the ultraviolet erasable EPROM operates is now presented. The actual memory element is an insulated-gate metal-oxide-semiconductor field-effect transistor. The conduction state of the device is controlled by the gate. If the transistor is biased in the conduction mode, one logic state is asserted, and if it is biased in the nonconducting mode, the opposite logic state exists. Through manufacturing methods, the gate may be isolated as well as insulated. The insulation makes the impedance extremely large. The isolation plays another significant role, as you will soon learn. There is an inherent capacitance present because of the physics of the device (two plates separated by a dielectric). Through the use of various manufacturing techniques, this capacitive effect may be exaggerated. The presence or absence of charge on this capacitor determines in which state of conduction the transistor is operating. Once the capacitor is charged, the isolation provides no discharge path. To program the device, high voltages are used to break down the insulating dielectric and charge the capacitor. Eight capacitors attached to eight gates determine the program pattern of one word or byte. The state of charge on these capacitors is determined by the programming word, and only those gates whose

state of conduction must be altered are altered. Once programmed, the device has a very long retention of the data pattern. Statistics show that in most cases the EPROM does not lose its pattern over the lifetime of the equipment in which it is operating.

The method used to erase the EPROM is to expose the device to high-energy short-wave ultraviolet light through its specially constructed quartz window. The energy from the radiation breaks down the dielectric and allows the capacitor to lose its charge. All this takes place in a controlled environment in order to prevent damage to the EPROM. The time period for the process is about 20 minutes for currently available devices. Once erased, the EPROM can be reprogrammed. If equipment using the EPROM is exposed to high-intensity UV radiation and no protection to the device is provided, failure in operation is to be expected.

Programmable EPROMs

Many types of PROMs may be programmed with the monitor development system through the use of the P command. Prior to using the P command, the EPROM programmer software must be entered into RAM memory for execution. Preferably, this should be done from cassette tape using the R command. Two things must happen for a successful load. The jump instruction with appropriate address must be entered into 401BH through 401DH, and the object code for the programmer must be entered, starting at the address of the jump. Three EPROM programming routines are presented in this chapter along with the necessary hardware that will allow use of these programs.

The first EPROM described is the 2780 multivoltage device, which was introduced by Intel Corporation. These EPROMs are contained in 24-pin dual in-line packages and require three voltages to operate (+12, +5, and −5 V). In addition, a +25 V programming pulse is required in the programming mode. This EPROM is second-sourced by many manufacturers. The data storage capacity of the EPROM is 1024 words of 8 bits each, or 1K × 8. A second device with larger memory storage was introduced by Texas Instruments Inc. (TI), the TI 2716. Programming it is the same as programming Intel's 2708. In this case, the storage capacity is 2048 words, or 2K × 8.

Intel also produces a 2K EPROM, but it is a single-voltage EPROM. This EPROM still requires a high voltage to program it, but the process is much simpler. A programmer for these devices is included in this chapter, and the details are presented in the section devoted to the programming process. Intel chose to call this EPROM a 2716. (This is not a printer error, the number is the same!)

At the same time, they introduced a +5 V dc 1K EPROM, calling it a 2758. These are most likely rejects from the 2716 processes with only half of the unit functional. A similar case occurred with the yields from the 2708 devices, producing the 2704 EPROM, a 512-byte version of the 1K device. (In any event, one must now be careful to note the manufacturer of the EPROM, as well as the type number of the device.)

Texas Instruments also supplies a single-voltage 2K EPROM, the 2516. Both the 2716 and the TI 2516 devices are pin compatible and available in a 24-pin dual in-line package. The memory capacity of EPROMs keeps expanding, and 4K single-voltage versions are available from many sources.

The last device considered in this chapter is more than an EPROM. In addition to the programmable memory, the device includes two input and/or output ports. It is directly compatible with the 8085 and 8748 microprocessors. This multipurpose IC is the 8755 and is manufactured by Intel Corporation. With a little "hardware" overhead, it can function on a Z-80 system such as the TRS-80. However, it is assumed that the 8755 will operate in an 8085 system such as the three-chip set described in the 8085 literature. The memory capacity of this device is the same as the capacity of the 2716. The programming of this EPROM is another variation of the technique used for the Intel 2716, but in addition there is sharing of address and data lines. Another difference exists in the packaging of the 8755, a 40-pin dual in-line device. It requires a special programming socket to accommodate the larger size.

Fig. 4-1 shows the pin configurations for the various EPROMs that can be programmed by the monitor system. The software packages given herein can program all of these devices. The user may wish to modify the software and hardware to program the 2732 (a 4K version of the 5 V 2716 EPROMs). But, there are critical differences in the pin functions between these two EPROMs. Also be aware that the program pulse is active low on the 2732 EPROM. A slight modification is necessary to program the Intel 2758 or TI 2508. The pin for the TTL programming pulse is changed, and a software correction is required. Refer to the programming specifications from the manufacturer if it is necessary to program this device.

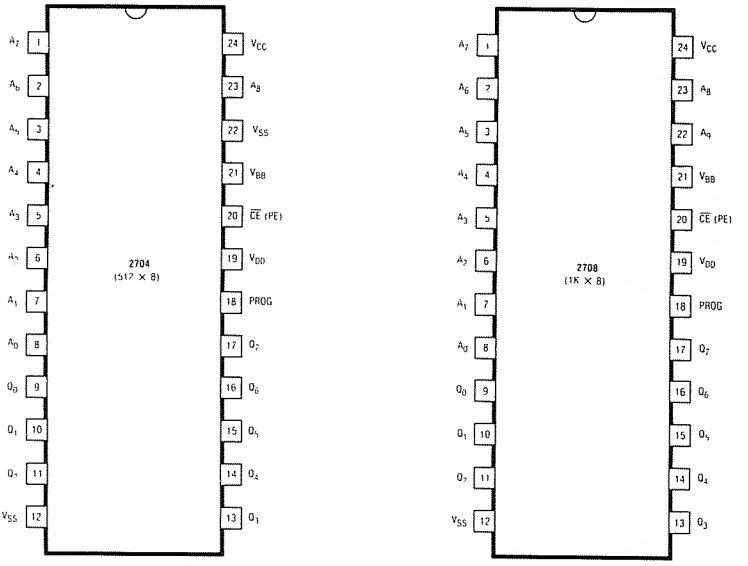
PROGRAMMING CONSIDERATIONS

Before the hardware is considered, a brief presentation of the programming of these three families of EPROMs must be presented. The programming methods as well as the number of voltages required to operate the device do influence the hardware design. The

one item common to all EPROMs presented in this text is their erased state, FFH in all locations.

The programming of the multivoltage devices is substantially different from the programming of the single-voltage EPROMs. This is true whether the device is a 1K or 2K byte EPROM. A more complicated procedure is required for the multivoltage devices, since these devices must be programmed sequentially in large blocks. The blocks are programmed many times in order to guarantee adequate programming. The exact process for programming may be obtained from the specification sheets provided by the manufacturer of each particular EPROM. For the most part, Intel specifications are used in the programs presented. The following sequence is necessary in the programming of the 2708 family of EPROMs.

The procedure for a program sequence starts with applying +12 V dc to the PROG/CS pin. The voltage at this pin remains high during the entire programming process. The address of the location to be programmed is placed on the address lines, followed by the data byte on the data lines. Some settling time is required so that all the signals can become stable before the programming pulse is applied. Twelve μs is a sufficient time, and the short delay caused by the instruction fetch and execute time fills this requirement. Next, a programming pulse is applied to the programming pin. The amplitude of this pulse is +25 V dc, but its width may vary according to



(*Continued on next page.*)

Fig. 4-1. Types of EPROMs that can be programmed by the monitor.

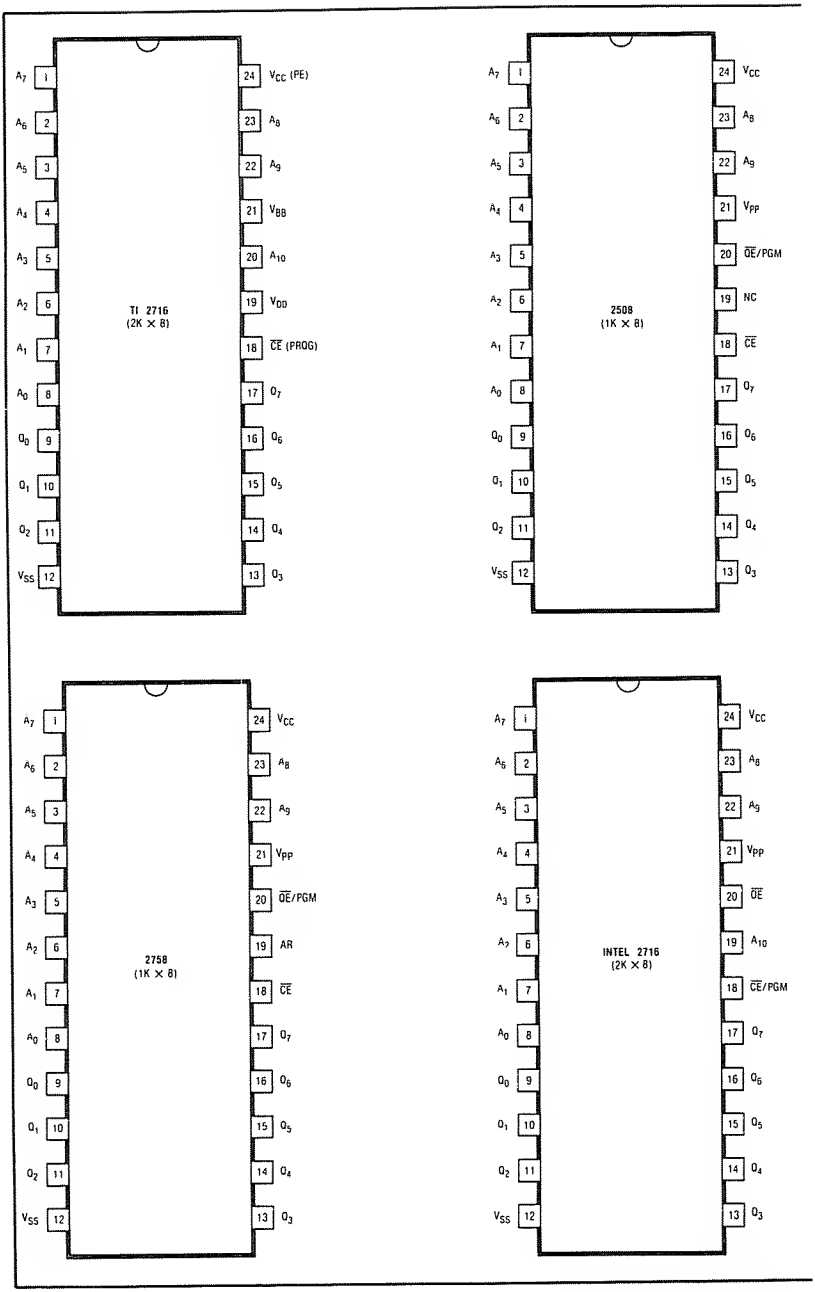
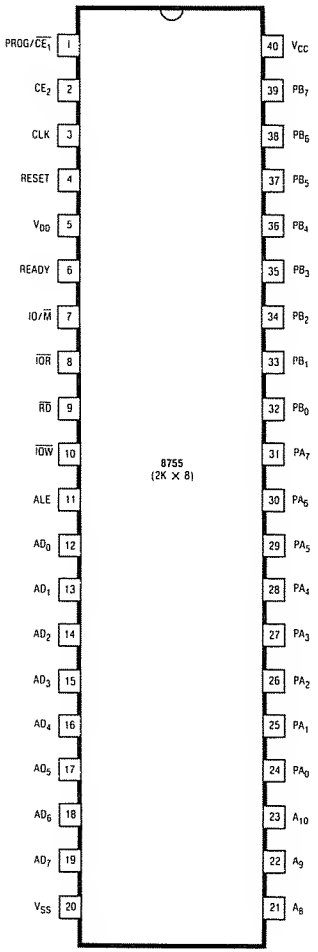
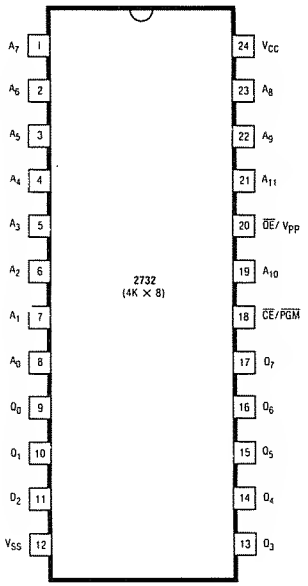


Fig. 4-1 (cont). Types of EPROMs that





Pin Names

AD_0-AD_7	Address Data Lines
A_0-A_{11}	Address Lines
O_0-O_7	Data Lines
CE	Chip Enable
OE	Output Enable
ALE	Address Latch Enable
AR	Select Reference Level
PGM	Program
PROG	High Voltage Program Pulse
V_{PP}	High Voltage Programming Voltage
PE	Program Enable (+12 Volts)
V_{CC}	+5 Volts
V_{SS}	Ground

Program functions refer to text

can be programmed by the monitor.

a predetermined relationship. (Intel specifies +26 V dc, but other manufacturers specify +25 V dc. The 25 V specification was used in the design of the programmer described.) The pulse may range from 100 μs to as long as 1000 μs. An empirical relationship between the number of program cycles and the width of the pulse has been established by Intel. (A program cycle is the time required to address and program all the locations within the EPROM.) If the program pulse is too wide, damage caused by overheating a location will result. If the pulse is too short, breakdown of the dielectric cannot be guaranteed, and an infinite number of cycles could not program the device. Hence, the two extremes are established.

A wide pulse allows a greater charge to be passed to the isolated capacitor. Hence, programming should occur with fewer passes. Conversely, the shorter pulse width allows a lesser charge to flow to the capacitor. Thus the relationship between the width and the number of passes is established. The product of the number of passes and the pulse width must be at least 0.1 second to ensure that proper programming has occurred. For example, if the width of the pulse is one ms, then 100 passes would be the minimum number to ensure proper charge on the capacitor. Extending the number of passes does not ensure a better programmed word, since the charging of a capacitor is governed by an exponential relationship, and a large number of addition cycles provides only a slight increase in the charge. The charging of the capacitor could come about using a shorter pulse, in which case more passes would be required. If the pulse width is reduced to 100 μs, then 1000 passes would be required to obtain the same effective charge as that obtained with the one ms pulse width. These two extremes define the permissible programming time for the EPROM. Stated mathematically,

$$N \geqq 0.1/twp$$

where,

N is the number of program cycles,
twp is the width of the program pulse.

The program sequence requires incrementing the address, applying the new data byte to be programmed, and then turning on a program pulse. This process must be repeated N times before the programming has been properly accomplished. Programming all locations of a 2708 device takes approximately 2 minutes.

If partial programming of a 2708 device is required, the data locations to be programmed are loaded in the usual manner, but the program cycle must artificially introduce a time period to compensate for the time absent from the program loop because not all locations are being programmed. This prevents damage to the programmed

device. One method used to create the delay is to program the same data that exists in the chip, changing only those locations in which the data is to be altered. This may be accomplished by transferring the data existing in the EPROM into a buffer area in spare memory, altering the locations in which changes are to be made, and then proceeding to program the entire block as if it were the first time that the EPROM was being programmed.

SINGLE-VOLTAGE EPROMS

The programming of the single-voltage devices is a much simpler process. In addition to the hardware differences, the controlling software is dramatically changed. The significant difference is that any location can be altered, providing the new pattern is compatible with the old pattern. The programming may be done in a random fashion. To program the 2716, the following sequence is performed in the programming algorithm. To initiate the program mode, a +25 V dc programming voltage is applied to the program pin. The high voltage is not pulsed but is constant. This voltage must be applied after power-up (that is, the +5 V dc is applied to the chip first), and it must not make a transition from 0 to 25 volts. The only allowable transitions are +5 V dc to +25 V dc and +25 V dc to +5 V dc. It may be only the Intel device that does not allow this transition, but in the hardware and the software presented, only the 5-to-25 and 25-to-5 volt transitions are possible. The device can be programmed in any order, but the new data pattern must not conflict with the data to be written over. This means that a zero may be programmed in any location where a one exists in the byte pattern, but a one cannot be programmed over a zero. The programming is accomplished with a single 50 ms, +5 V pulse on the V_{pp} pin.

If the system microprocessor is 8080 compatible, it may be desired to program 00H in all unused locations, since a 00H is a no operation (NOP) instruction.

8755 EPROM I/O CHIP

The 8755 is not programmed in the same fashion as the Intel 2716 device. The random, single-programming pulse technique is used, but in this case, the pulse is on the +25 V dc line. Other changes include a large socket holder that must now hold a 4-pin package, and shared data and address lines. In the programmer design, the EPROM is connected to the common function pins of the 2716, except that the only address lines connected are the high-order bits 8, 9, and A. The data and the low-order address share the same pins.

This is accomplished with the address latch enable (ALE) signal, which is not directly available with a Z-80 based system, but may be simulated with the software to allow both the programming and verification of data. Other lines support the memory and input-output functions.

EPROM PROGRAMMER HARDWARE

The hardware used to program the EPROMs uses an 8255 programmable peripheral interface (PPI) chip. This chip is available from many sources at a reasonable cost. While it is possible to build the programmer for under $20 if zero-force insertion sockets are avoided, you would be wise to use these sockets to avoid damaging the expensive EPROMs. In connecting the EPROM programmer to the TRS-80 bus, the address lines are buffered, but the data lines are not. Separate sockets for 2716, 2708, and 8755 EPROMs are used in the design. You must be extremely careful not to plug a single-voltage device into a multivoltage socket. Also, be aware that the programming circuitry is connected to all device sockets, so only one EPROM device should be installed for any one programming session. Violation of this rule will result in permanent damage to the EPROM. *BE CAREFUL!*

Fig. 4-2 outlines the circuit of the EPROM programmer, and Figs. 4-3 through 4-6 show details for specific EPROMs. The +25 V dc source is shared by all devices. To program 2708 devices (Fig. 4-3), this voltage source must be able to sink several milliamperes. This explains the presence of a special active current sink in the design. The single-voltage 2716s and 8755s require that this voltage go from +5 to +25 V dc, instead of from 0 to 25 V, so the circuit design reflects this requirement (Figs. 4-4 and 4-5). The difference in the application of these voltages is determined by software, and is dependent upon which device is to be programmed. For example, the 2716s allow this voltage to be constant while the EPROM is being programmed, whereas the 8755s require that the voltage supplied be a programming pulse of 50 ms duration. Yet another requirement exists for programming the 2708s, since these require 0 or +12 V dc to be applied to the chip select/program enable line (CS/PROG) instead of the normal TTL levels of 0 and +5 V dc.

To program the Texas Instruments TI2716s, which you recall are programmed in a manner similar to the 2708s, more hardware changes are needed (Fig. 4-6). The change shown allows the selection of either the upper or lower IK sections of the EPROM. With this modification, the software routines supplied can program the TI device, providing it is done in two sections.

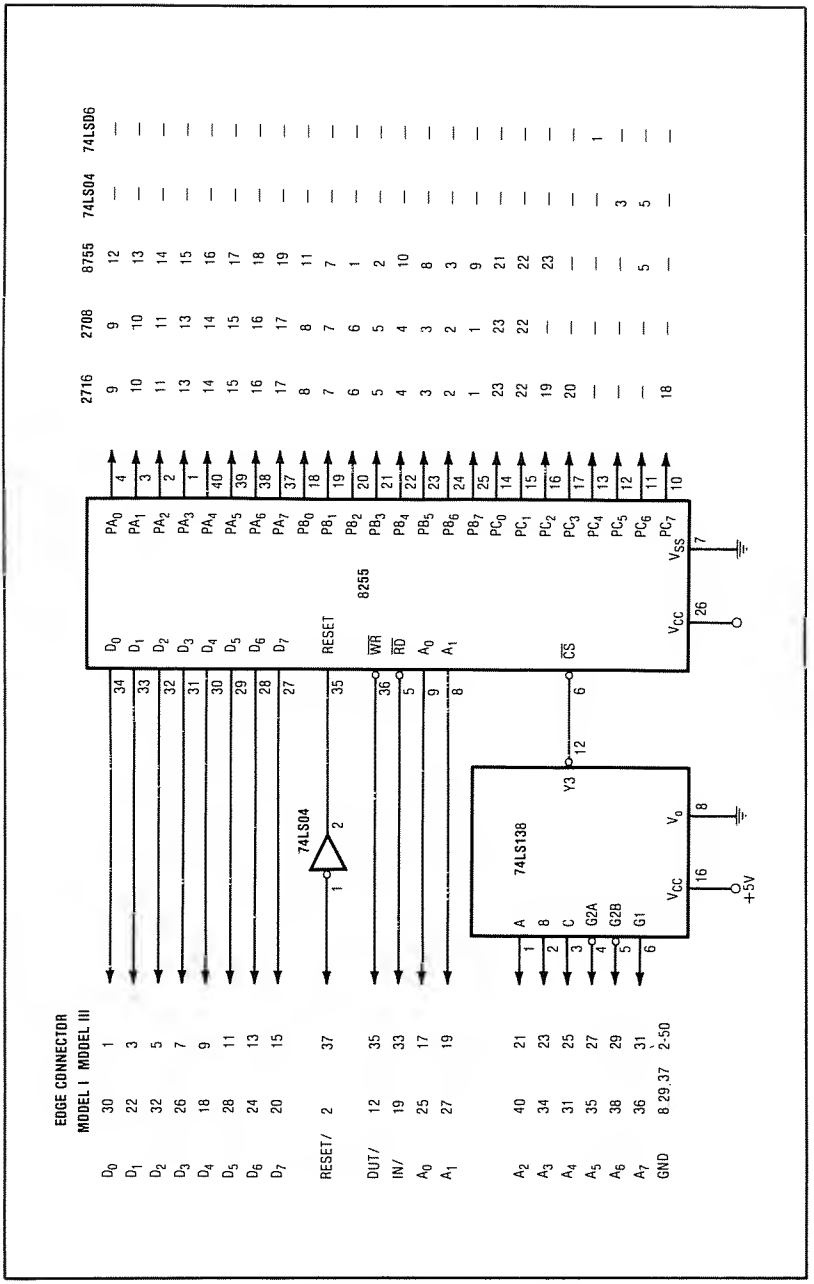


Fig. 4-2. The basic EPROM programmer.

If you have no intention of programming a particular EPROM, the implementation of the corresponding hardware is not necessary. The major feature to consider is the flexibility that this particular EPROM programmer provides.

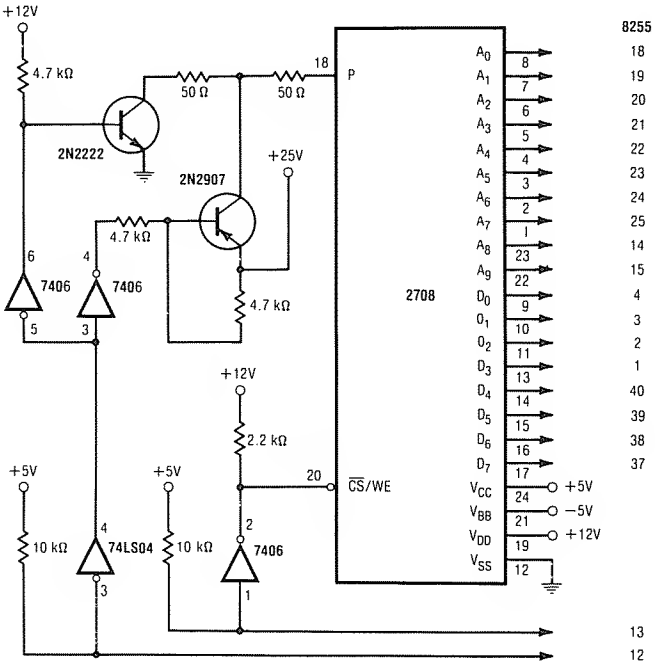


Fig. 4-3. 2708 EPROM programming circuitry.

The flexibility of the EPROM programmer is provided by the 8255 PPI, a very powerful device. A detailed explanation of this chip is available from the specification sheets. Another excellent reference is Goldsbrough's book[1], which is devoted to the 8255 PPI interface chip. The monitor software utilizes mode 0 (not to be confused with interrupts) for programming, reading, and verification. Since a particular port on this chip may be either an output or an input (a function of the control word used to initialize the device), data may be written on the data lines for programming and read from the data lines for verification. Also note that the RESET line is active high. At power-up it is necessary that all ports be configured in the input state, producing a high-impedance state at their corresponding pins. Pull-up resistors guarantee that no high voltages are applied to an EPROM socket while in the initialization process. It

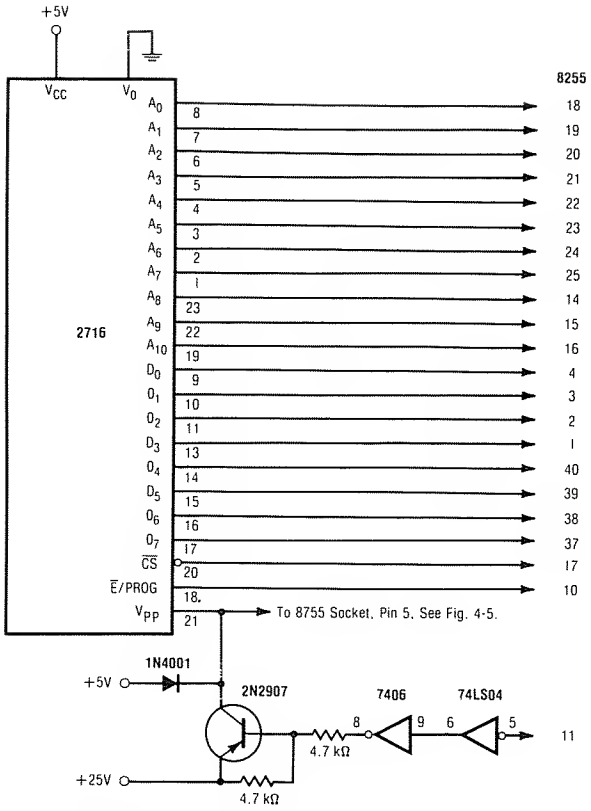


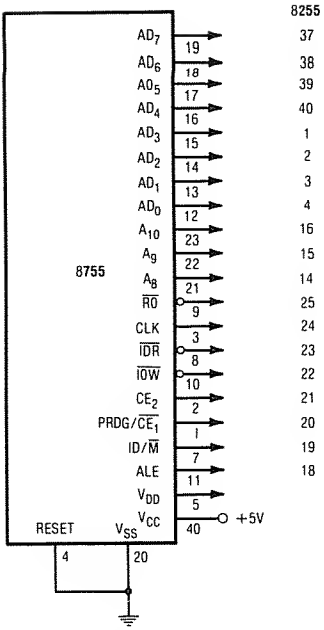
Fig. 4-4. Circuitry used to program the 2716.

is best to keep all EPROMs out of sockets until it is time to program them.

The following port assignments are made on the 8255 and can be verified by observing the circuit diagram. Port A is used for data and changes from an input to an output depending upon the function and the use of the appropriate control word. Bits 0 through 7 are connected to the data lines of the 2708 and 2716 EPROMs. The monitor addresses this port with an 8CH. When programming the 8755s, these lines are used for both data and address information. You may observe this by noting the connections on the 8755 socket.

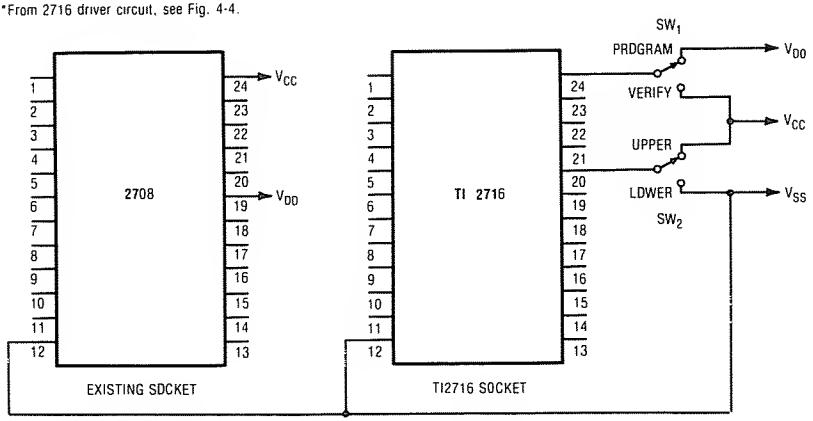
Port B, addressed by the monitor with 8DH, is used for low addresses in both the 2708 and 2716 devices, but not for the 8755. Port A is used in this case.

Port C, addressed as 8EH, is used for both address information and control in the programming of all EPROMs. The four least sig-



*From 2716 driver circuit, see Fig. 4-4.

Fig. 4-5. Programming circuitry for 8755 EPROM I/O.



PIN 18 PULSE = 25 V DC

Note: All pins except pins 20 and 24 interconnect directly to equivalent locations from the 2708.

Fig. 4-6. Programming the TI 2716 with 2708 circuitry.

nificant bits are used for addressing. The remaining bits are used for control purposes that provide the program pulses or voltage levels. The function depends upon which device is to be programmed. However, the four most-significant bits have dedicated EPROM

device functions. The specific allocation of each pin is given in the software-hardware section for programming a particular class of EPROM. The last port address, port 8FH, is used to supply the control word that determines the port configuration of the 8255.

There is a general method of implementing the EPROM routines. Depending upon which EPROM family is to be programmed, the particular software to drive the programmer must be entered in memory. These software routines are provided in the sections covering each EPROM family type. If relocation is necessary, it is best that it be done with an editor/assembler. After entry and verification of the object code, the program may be stored on cassette tape. This is done using the W command. How the jump vector is loaded into locations 401BH to 401DH is a matter of choice. The vector may be entered using the monitor at the time the tape containing the EPROM program is loaded. An alternative is to produce a tape containing the jump information using the W command to save the three bytes that form the jump. Assuming the object code to be used for programming an EPROM is loaded at 443FH through 47B4H and the code to support the jump vector is entered at 401BH through 401DH, execution of the following statement on the monitor saves the jump vector and the program on cassette tape for future use:

```
W401B 3 0066 PSTART,W 443FT47B4 0066 EPROM1<ENTER>
```

The execution address is always loaded into the program counter. A safe re-entry into the monitor at location 0066H is the same as pushing the reset push button. When it is necessary to program a specific family of EPROM, the tape is read using two R commands:

```
R,R<ENTER>
```

After loading the program, executing the P command causes the monitor to execute the EPROM programming software. All the reading, verification, and programming options are then available.

In the presentations that follow dealing with the EPROM families, the first section gives the pin assignments, and a second section lists the source code to be used to "drive" the EPROM programmer.

PROGRAMMING THE 2708 EPROM

To program the 2708 EPROM it is necessary to supply addresses, data, and control pulses to the device. The addresses are latched at ports B and C on the 8255. The 2708 is a 1K EPROM, so ten address lines are required (A_0 through A_9). The low byte address lines defining the location of the data word to be programmed are at-

tached to port B bits 0 through 7, which equate to address lines 0 through 7. Addresses A8 and A9 are connected to port C, bits 0 and 1. Bits 2 and 3 are not used in this programming module. Although not documented in this section, these bits could be used for programming the TI2716 if appropriate hardware and software changes are incorporated. (Note that it is possible to program the TI2716 with the hardware presented using the modification shown in Fig. 4-6. This modification treats the TI2716 as two 1K sections with each section programmed independently, as previously noted. A switch controls which block is addressed by the EPROM programmer.) Bits 4 and 5 of port C are used for control of dc levels and for the programming pulse. Chip select and program (CS/PROG) pin 20 of the 2708 is controlled by bit 4. In the programming process, this terminal must be driven at +12 V dc to provide the program enable (PROG) function. To accomplish this in the design, an open-collector TTL device (74LS06) is used with pull-up resistors to +12 V. Bit 5 of port C supplies the programming pulse that is applied to pin 18 of the 2708. The duration and the number of programming loops are determined by the software. The data is applied to the 2708 through port A. During reading and verifying the programmed bytes, this port changes directions and becomes an input port, and the data is read by the monitor through this port. The control word of the 8255 determines the direction of data flow. The remaining pins of the 8255 are not used in the programming, reading, and verifying of 2708 EPROMs. These bits are masked with the software to appropriate levels to prevent possible damage to other devices when the 2708 command function is executed.

The sequence for programming the 2708 requires the presence of a block of data of 1024 bytes. If you must program only a few locations, transfer the contents of the EPROM to a buffer, and change the locations in question. Remember that a 2708 location that contains FFH can be programmed to any pattern, and that only the 1s in a bit pattern can be programmed to 0s. Thus, it is possible to "write over" a programmed location without first erasing the EPROM, but only logic 1s may be changed to logic 0s, and not vice versa. With this technique, you can program any number of bytes. The program supplied uses this method for programming shorter blocks.

In the development of the software used to program the 2708s, the following procedure is used: Port A is configured as an output port for data. Port B and port C, bits 1 and 2, are also configured as output ports, and they hold the address information. The address of the EPROM location to be programmed is transferred to the 10 bits of the designated address ports. Port A is then set with the data word

to be programmed. The high-order bits of port C perform the following functions. First, bit 4 is set high, enabling the write enable line (PROG). Next, after a delay of at least I2 μs, which allows the data and address lines to stabilize, the programming pulse is activated by writing a I to bit 5 of port C, while keeping all other bits unchanged. This write pulse is maintained for a time that may vary between 100 and I000 μs. This is a function of the processor clock; the software must reflect the correct time delay for the Model I or Model III. This wait period defines the pulse width of the programming pulse. Bit 5 is reset after the selected pulse, and all other bits must remain unchanged. The programming procedure is repeated for the next EPROM address until all desired words have been programmed. At this point, one loop has been programmed, and the process must continue over again until the loop requirement is satisfied, that is, until each EPROM location has been programmed the correct number of times (corresponding to the ratio of the total program period to the pulse width period, which equals the number of loops).

Using the EPROM Programmer

After the program to be put in the EPROM is available on cassette tape, available in another EPROM, or entered using the monitor, the programming session can start. Be sure the software contains the correct time delay for the model in use. If the program to be put into the EPROM is on tape or is to be entered using the monitor, the program should be placed in memory above 5000H. The EPROM programmer uses most of the R/W memory below 5000H. If the data to be programmed is available in a compatible ROM or previously programmed EPROM, then it can be entered using the R command described below.

Five commands are possible in the EPROM programmer mode. All commands are interactive; that is, return to the monitor functions occurs only if requested. To terminate a command without returning to the monitor, use the <SHIFT> key. The programmer message:

```
2708 EPROM PROGRAMMING PROGRAM
>
```

appears on the crt screen. This is the same message that appears when the P command of the monitor is used.

Read Command

The first command is a read (R), in which the contents of an inserted ROM or EPROM are transferred to the addresses specified by the user for possible review and modification. The first field deter-

mines the placement of the data read from the ROM or EPROM, and the second field determines the number of data bytes to be read. The third field is an offset in the EPROM start address. If you wish the first location in the EPROM to be loaded in the first RAM location specified, this field may be omitted in the command:

```
R 5000T53FF 000<ENTER> or, R 5000 400 000<ENTER>
R 5000T53FF<ENTER> (no offset)
R 5000T52FF 100 <ENTER>
```

The data in the EPROM locations specified is transferred to memory starting at address location 5000H. If no offset is given, or if it is 000H, the data in the first location is transferred to 5000H. If an offset is present in the command, the transfer from the EPROM starts at the offset address and continues until the entire block length specified is transferred.

Verification Command

The second command allows verification (V) of all or part of the data within the EPROM. The data to be verified must be loaded into the monitor from a cassette tape, another ROM or programmed EPROM, or code already present in R/W memory. Assuming that a previous read command loaded the data to be verified with the contents of the EPROM, then:

```
V 5000T53FF 000<ENTER> or, V 5000 400<ENTER>
```

compares the data in the EPROM with the data at memory locations 5000H through 53FFH. Any discrepancies appear on the crt screen. The format shows the address and data in the memory followed by the address and data in EPROM. For example the crt display shows

```
MEMORY: XXXX XX  PROM: XXXX XX
        XX DESCREPANCIES
```

for each discrepancy, and terminates the list with the total number of differences in the range indicated. Any block length can be specified.

Programming Command

The third option is the programming (P) command. Any number of data bytes can be programmed. The data to be programmed is entered by one of the methods described for the verification command. The programming time is approximately 2 minutes and is independent of the block length. The format for this command is

```
P 5000T53FF 000<ENTER> or, P5000 400<ENTER>
```

which takes the data present at locations 5000H through 53FFH and places it into EPROM locations 0000H through 03FFH. As with the read and verify commands, it is not required to start at the first location in the EPROM. For example, assume you wish to program locations 0050H through 014FH of the EPROM with the data located from 5230H through 532FH. The format to perform this programming function is as follows:

```
P 5230T532F 50<ENTER>, or P5230 100 50<ENTER>
```

The programmer automatically verifies the data in the EPROM with the data in R/W memory. Any differences are shown using the same format as that used in the verification mode. It is possible to program only one byte of a 2708 EPROM, but the time required is the time of programming all locations within the EPROM.

Short-Cycle Program Command

You may also request the short-cycle (S) option by using the fourth command, which programs the EPROM and then verifies it after each programming loop through the block being programmed. In this way, time is saved, since all of the loops may not be required to program the EPROM successfully. To be sure that the EPROM has been completely programmed, the short-cycle program reprograms the EPROM for several additional cycles. This programming method may save you some time during program development, but it does not meet the manufacturer's specifications and should not be used for the final programming of the EPROM that will be used in a product or application. To use the short-cycle programming, the following command is used:

```
S 5000T53FF 000<ENTER> or, S 5000 400<ENTER>
```

This command performs the programming function of transferring the data at memory locations 5000H through 53FFH into EPROM locations 0000H through 03FFH. Programming continues until verification shows no errors. This produces a programmed device in less than 30 seconds when a new 2708 EPROM is used.

Exit Programmer Command

The final command,

```
E<ENTER>
```

allows return to the FROLIC monitor for the monitor functions.

Program Locations for the 2708 EPROM

The software shown in Appendix F is annotated in the hope that you can follow the programming sequence and add or remove code

as you deem appropriate. The address to be placed in the jump vector locations 401CH and 401DH is 443FH. Remember to place the address in the Intel format of low byte followed by high byte.

PROGRAMMING THE 2716 EPROM

The hardware configuration for the single-voltage 2716 EPROM is similar to that for the 2708. Port B is used for the lower address bits, and port C bits 0, 1, and 2 are used for the additional addresses. This makes a total of 11 addresses, and, therefore, 2048 locations can be addressed. Port A is used for data, and it can be programmed to write or read depending upon the function desired. Bit 3 of port C is used for the chip select (CS/) at pin 20. Unlike the 2708, it is not necessary to raise the level of this pin to +12 V dc to program the EPROM. Only one high voltage is required, which is not pulsed but is constant. Control of this voltage is obtained from bit 6 of port C, which raises pin 21 of the 2716 to +25 V dc when programming is to start. The actual programming of a word into the EPROM takes place using a single TTL-level pulse of 50 ms duration on the programming pin (PROG), which is pin 18 of the 2716. This pulse is output from bit 7 of port C. The two remaining bits of port C are used to program 2708 EPROMs. Software is used to control the address, data, and pulse width.

If you study the schematic diagrams (Figs. 4-2 through 4-4), you can see that the programming operations for the 2716 and the 2708 are separate. It is possible to write software that copies the contents of one of these PROMs to the other without the need of an intermediate buffer.

The sequence used for programming the 2716 is vastly different from the procedure used to program the 2708. The 2716 may be programmed in a random manner, and it is possible to program a single byte in 50 ms, a great improvement over the time required to program a 2708.

The procedure is as follows: the address is placed on the address lines using ports B and C. The program pin voltage of pin V_{pp} is brought to the +25 V dc level. It is possible to have done this first, since the data read operation for the verification of these devices can take place with this pin at the +25 V dc level. The data is written to port A as an output port, and a +5 V dc pulse is applied to the PROG pin for 50 ms, keeping all other bits unchanged. The +25 V may be removed, but if more data is to be programmed, it must be present until the entire programming sequence is finished. Since the address is still on the address lines after a location has been programmed, it is possible to verify the programmed data word by changing port A

from an output to an input. This is accomplished by using the appropriate command word. The +25 V dc line does not have to be changed to 5 V dc in this read operation.

This completes the programming process with the data word having been written in the location specified by the address lines. This data can be removed only through the application of high-intensity ultraviolet light. The erasure process is not as selective as the programming process, and all bits are erased when the EPROM chip is subjected to the ultraviolet light.

The monitor commands used to program the 2716 EPROM are the same as those used to program the 2708. However, there is no short-cycle program command. After the EPROM programming program has been entered in memory, it should be saved using the W command. When the P command is executed, a message is displayed indicating the EPROM family that the software supports. In this case it would indicate that 2716s may be programmed. This is done to help you avoid programming the wrong EPROM family. The options allow for reading (R), verifying (V), and programming (P) of the total EPROM, as well as operating on single bytes. An example of programming a single byte is done using a block specification of 1, or

```
P 5135T5136 135<ENTER> or, P 5135 1 135<ENTER>
```

In this case, the data located in memory location 5135H is transferred to the 0135H EPROM location. Verification is automatically performed.

The program also allows programming the 1K, +5 V dc 2508 or 2758 EPROMs if a software change is made to apply the programming TTL pulse to pin 20 instead of pin 18. CAUTION! Damage to the 2508 or 2758 results if this change in software is not made. Refer to the Intel or Texas Instruments specifications for programming these devices.

The software shown in Appendix G is annotated with the hope that you can follow the programming sequence and add or remove code as you deem appropriate. The address to be placed in the jump vector location 401CH and 401DH is 4400H. Remember to place the address in the Intel format of low byte followed by high byte. When saving the program on tape, note the starting and ending addresses as shown in the listing.

PROGRAMMING THE 8755 EPROM

The next EPROM type that the EPROM programmer can support is the 8755 device. This integrated circuit contains not only 2K × 8

of memory, but also two input/output ports. The added ports make it necessary for the manufacturer to increase the size of the package to 40 pins. It also is necessary to share lines to provide all of the needed functions. Therefore, the data and low address use a common set of pins. This is made possible by the addition of the address-latch enable (ALE). (Note that this chip is not directly compatible with the Z-80 microprocessor because of the shared lines. It is possible to adapt the 8755 so that it could operate on the TRS-80 bus. However, it is being assumed that this chip is to be used in a special dedicated microprocessor controller which was developed with the TRS-80 development system to run on an 8085 system.) Because of the multiplexed address and data bus, the programming of the device is complicated slightly. In this adaptation, the 8255 is also used to access the input-output control pins of the 8755. More information on this chip may be found in the 8085A Cookbook[2] or in application literature on this product.

The procedure used for the programming of the 8755 EPROM is as follows: the low address is placed on the data lines, using PPI port A, and is latched into the 8755 through the use of the ALE input, pin 11. Port B, bit 0, is used to control this line. All addresses must be present when the latching action occurs. This includes the high address bits (A8, A9, and A10) that are obtained from port C bits 0, 1, and 2. The ALE line is high or goes high during the address set-up. The input-output memory (IO/M) line is also latched in at this time. This line is brought low using port B, bit 1. A low on the IO/M line establishes the device as memory. The two chip-enable lines, CE1/PROG, pin 1 (active low and also functions as the program select when active high) and CE2, pin 2 (active high), must also be asserted at the time the address is set up. Port B, bits 2 and 3, control the chip select lines. When the above conditions are valid (address on address lines, chip selects active, and IO/M low), the ALE is brought low, which latches both the address and the memory logic states in the 8755. Once this data is latched, the PROG/CE1 may be activated, changing it from a low to a high, establishing the program mode. The PROG pin must remain high until after the programming pulse has been applied. Data is placed on the data lines of port A. After a delay of 2 μs, V_{dd}, pin 5 (driven by port C, bit 6 of the 8255), is pulsed from its +5 V dc level to +25 V dc for 50 ms, producing the programming pulse. The 8755, like the 2708, requires that the high voltage be pulsed to program the word. However, unlike the 2708, only one pulse is required for programming. The PROG/CE1 input should remain high for an additional 2 μs after a word is programmed. If desired, a verify or read (RD/) can be initiated to check the validity of the programmed word.

Port A of the peripheral chip must be reversed from output to input using the control word. The address is still latched because the ALE has remained low during the programming sequence. Therefore, all that is required after the data flow is reversed is to assert the read line (RD/) using port B, bit 7. The data is read from the peripheral chip by reading the value from port A. If it is the same as the word programmed, verification is complete. As with the 2716 EPROM, data may be programmed in a random fashion. It is also possible to write over data, if only logic 1s are to be changed to logic 0s. The opposite is not possible without a complete erasure of the EPROM.

The software shown in Appendix H is provided to support the programming of the 8755 devices. The commands are the same as those used in the other EPROM programmers. When the P command is executed, a message informs you which device this program supports. The commands are read (R), verify (V), and program (P).

The command format for programming the 8755 EPROMs is identical to the formats for the 2708 and 2716 EPROMs. The software is loaded by any convenient method, then stored on cassette tape.

The software shown in Appendix H is annotated with the hope that you can follow the programming sequence and add or remove code as you deem appropriate. The address to be placed in the jump vector location 401CH and 401DH is 4400H. Remember to place the address in the Intel format of low byte followed by high byte. When saving the program on tape, note the starting and ending addresses as shown in the listing. Remember to include the jump vector when originating this tape. Reference should be made to the section dealing with the 2708 EPROM.

REFERENCES

1. Goldsbrough, P. F. *Microcomputer Interfacing with the 8255 PPI Chip.* Indianapolis: Howard W. Sams & Co., Inc., 1979.
2. Titus, C. A., Larsen, D. G., and Titus, J. A. *8085A Cookbook.* Indianapolis: Howard W. Sams & Co., Inc., 1980.

APPENDIX A

Command Sequence Table

NOTE: Most <ENTER>s may be substituted with (,) (s) and commands may be sequenced. The only exception is the T command.

Command	Function	Command and Parameters
@	Fill memory with a constant over specified range.	@<LOW ADDRESS> T <HIGH ADDRESS> <CONSTANT> <ENTER> @<LOW ADDRESS> <HIGH ADDRESS> <BLOCK LENGTH> <CONSTANT> <ENTER>
A	Display data over specified range, ASCII representation of hex data.	A<LOW ADDRESS> T <HIGH ADDRESS> <CONSTANT> <ENTER> A<LOW ADDRESS> <HIGH ADDRESS> <BLOCK LENGTH> <ENTER>
B	Execute saved command buffer.	B<ENTER>
BS	Save present commands in buffer.	BS<ENTER>
C	Clear screen.	C<ENTER>
D	Display data in hexadecimal format over specified range. The address is at the beginning of each line, followed by the data in it. Up to sixteen memory addresses are displayed.	D<LOW ADDRESS> T <HIGH ADDRESS> <ENTER> D<LOW ADDRESS> <HIGH ADDRESS> <BLOCK LENGTH> <ENTER>
E	Execute starting at address specified. Current screen data is used at start of execution. When a breakpoint is	E<ADDRESS>.<BREAKPOINT> <ENTER> E<ADDRESS>.<BREAKPOINT>. <BREAKPOINT> <ENTER>

Command	Function	Command and Parameters
	reached, the data residing in the current screen locations is transferred to the memory locations specified by the V(ideo) pointer.	E<ENTER> (Current Program Counter is used for execute address.)
F	Find single data byte over range specified. Display address of each occurrence on crt screen.	F<LOW ADDRESS> T <HIGH ADDRESS> <DATA> <ENTER> F<LOW ADDRESS> <HIGH ADDRESS> <BLOCK LENGTH> <DATA> <ENTER>
G	Execute starting at address specified. The data in the crt R/W memory is replaced by that stored in the memory locations referenced to by the V(ideo) register. When a breakpoint is reached, the data in the crt locations is stored at the addresses specified by V.	G<ADDRESS>.<BREAKPOINT> <ENTER> G<ADDRESS>.<BREAKPOINT>. <BREAKPOINT> <ENTER> G<ENTER> (Current Program Counter is used for execute address.)
H	Hexadecimal sum and difference of two values. Value 2 is added to value 1; then value 2 is subtracted from value 1.	H<VALUE1> <VALUE2> <ENTER>
I	Insert object code or data into memory starting at address specified. Address may be changed using the /. The buffer is only 255 bytes long, and when it is full no further data is accepted. Deposit is accomplished by pressing <ENTER>.	I<ADDRESS> <DATA> <DATA>. . . <DATA><ENTER> I<ADDRESS> <DATA>. . .<DATA> <ADDRESS>/ <DATA>. . .<DATA> <ENTER>
J	Not used.	
K	Not used.	
L	Hard copy flag toggle for listing on printer. (Hard copy driver program must be loaded in R/W memory. See text.)	L<ENTER>
M	Move the contents of memory contained in locations specified to new locations starting at specified destination address.	M<LOW ADDRESS> T <HIGH ADDRESS> <DESTINATION ADDRESS> <ENTER> M<LOW ADDRESS> <HIGH ADDRESS> <LENGTH> <DESTINATION> <ENTER>
N	Input data from input port or range of ports.	N<PORT NUMBER> <ENTER> N<PORT NUMBER> T <PORT NUMBER END> <ENTER> N<PORT NUMBER> <NUMBER OF PORTS> <ENTER>

Command	Function	Command and Parameters
O	Output data to port specified.	O<PORT NUMBER> <DATA> <ENTER>
P	Program EPROMs. (It is required that this option is available in R/W memory. See text.)	P<ENTER>
Q	Search for two-byte word in Intel format, low byte followed by high byte. Each occurrence is displayed.	Q<LOW ADDRESS> T <HIGH ADDRESS> <DATA> <ENTER> Q<LOW ADDRESS> <HIGH ADDRESS> <BLOCK LENGTH> <DATA> <ENTER>
R	Read cassette tape using TRS-80 500 bits per second format. A bias may be entered, and the value of this bias is used to offset the placement of the data in memory from that specified in the format.	R<ENTER> R<BIAS> <ENTER>
S	An alternate form of substituting object code or data into R/W memory. This command is interactive. A double <ENTER> returns to buffer sequence execution.	S<ADDRESS><ENTER><DATA><ENTER> <SPACE><ADDRESS>/<DATA> <ENTER> <DATA> <RUBOUT> <REPLACEMENT DATA> <ENTER> <ENTER>
T	Transfer data in video storage to screen to show the state of the crt when a breakpoint was encountered. The V register must point to the starting location of the storage address. If V is 3C00H (default value), then the data is stored in the video R/W memory and will be scrolled. This command is useful when program development uses video screen locations for i/o.	T<ENTER>
U	User assignable. (Option must be loaded into R/W memory. See text.)	U<ENTER>
V	Verify locations specified in first locations specified with the data in locations starting at the address in the destination field.	V<LOW ADDRESS> T <HIGH ADDRESS> <DESTINATION ADDRESS> <ENTER> V<LOW ADDRESS> <HIGH ADDRESS> <LENGTH> <DESTINATION> <ENTER>
W	Write a tape cassette using Radio Shack format. Name is optional.	W<START ADDRESS>T<END ADDRESS> <EXECUTE ADDRESS> (<NAME>) <ENTER>

Command	Function	Command and Parameters
		W<START ADDRESS> <LENGTH> <EXECUTE ADDRESS> (<NAME>) <ENTER>
X	Examine and/or modify registers. Valid register names are as follows: P Program counter S Stack Pointer X IX Register Y IY Register N Interrupt Flag 1 set 0 not set I Interrupt Vector V Video screen storage pointer A, B, C, D, H, L, F Registers A', B', C', D', E', H', L', F' Alternate Registers	X<ENTER> (All registers displayed, no modifications.) X<REGISTER> <ENTER> (Only specified register displayed.) X<REGISTER> <DATA> <ENTER> (Data replaces former contents.)
Y	Not used.	
Z	Single-step function. Hardware must be installed for this command to function.	Z<ENTER> (Current program counter is used for start address.) Z<EXECUTE START ADDRESS><ENTER> Z.<BREAKPOINT><ENTER> (In case command executed is a disable interrupt, the breakpoint is used to return to monitor.)

APPENDIX B

References

1. *Zilog Z-80 Technical Manual.* Cupertino, Calif.: Zilog Corp., 1976.
2. *TRS-80 Microcomputer Technical Reference Handbook.* Radio Shack, 1978.
3. *TRS-80 Model III Operation and Basic Language Reference Manual.* Radio Shack, 1980.
4. McNamara, John E. *Technical Aspects of Data Communication.* Maynard, Mass.: Digital Press, 1977.
5. *Radio Shack Service Manual, TRS-80 Model III Microcomputer.* Radio Shack, 1980.
6. *Intel Component Data Catalog 1980.* Santa Clara, Calif.: Intel Corporation, 1980.
7. Goldsbrough, Paul F. *Microcomputer Interfacing With the 8255 PPI Chip.* Indianapolis: Howard W. Sams & Co., Inc., 1979.

APPENDIX C

Hardware and Software Supplier

An EPROM chip containing the FROLIC™ monitor is available for both the Model I and Model III TRS-80 computers. Due to differences in the computer circuits, the software is not exactly equivalent. However, the monitor functions are the same.

The source program is also available for both computers on a cassette tape. The monitor may be used with 16K, 32K, or 48K of read/write memory. A floppy disk containing the source program and the object program is available for computers with 16K, 32K, or 48K of read/write memory.

The electronic components, printed circuit boards, and other supplies for the single-step interface, the EPROM programmer, and the serial i/o interface are also available. Each kit or assembled unit is supplied with a cassette tape that contains the source program(s) for the interface. These program tapes may be purchased separately.

All of the hardware and software is available from:

Frolic Devices
P. O. Box 772
Bloomfield, CT 06002

A stamped, self-addressed envelope may be sent to Frolic Devices for current prices and delivery information.

APPENDIX D

Source Listing for FROLIC Monitor for the Model I TRS-80

```
                00010 ;
                00020 ;         MODEL I TRS-80 DEVELOPMENT SYSTEMS
                00030 ;         DISK NAME MODI/ASM
                00040 ;         MODEL I FROLIC MONITOR
                00050 ;
4012            00060 RSTLOC    EQU 4012H
00FF            00070 RST       EQU 0FFH
4000            00080 REF       EQU 04000H      ;START OF RAM
4080            00090 BUFFER    EQU REF+80H
4180            00100 TEM       EQU REF+180H    ;SAVED BUFFER AREA
4040            00110 REF1      EQU REF+40H
4040            00120 CURSOR    EQU REF1
4042            00130 SSFLAG    EQU REF1+2      ;SINGLE STEP FLAG
4043            00140 COUNT     EQU REF1+3H
4044            00150 CMARK     EQU REF1+4H
4045            00160 BPNO      EQU REF1+5H
4048            00170 BPOINT    EQU REF1+8H     ;BUFFER POINTER
404A            00180 PRNTFG    EQU REF1+0AH
404B            00190 PRIMF     EQU REF1+0BH
4024            00200 REF2      EQU REF+24H
4024            00210 SSTACK    EQU REF2
4026            00220 BOTMS     EQU SSTACK+2H
403C            00230 PCSAVE    EQU REF2+18H
403A            00240 HLSAVE    EQU PCSAVE-2
4080            00250 BREAKS    EQU BUFFER      ;JUST BELOW BUFFER
3C00            00260 VIDEO     EQU 3C00H
4047            00270 OLD       EQU REF+47H
4046            00280 UCFLAG    EQU REF+46H
403E            00290 SCREEN    EQU PCSAVE+2H
4018            00300 TTYP      EQU REF+18H
401B            00310 PROM      EQU REF+1BH
401E            00320 USER      EQU REF+1EH
4300            00330 USERSP    EQU REF+300H    ;USER STACK POINTER
42C0            00340 MONSP     EQU USERSP-40H  ;MONITOR STACKPOINTER
42A0            00350 JMP       EQU MONSP-20H   ;TOP OF MONITOR STACK
                00360 ;
0000            00370           ORG 00000H
                00380 ;
0000 AF         00390           XOR A           ;CLEAR AC
0001 D3FF       00400           OUT (0FFH),A    ;LOWER CASE AND CASSETTE OFF
0003 210040     00410           LD HL,REF       ;SYSTEM RAM
0006 1803       00420           JR S1           ;BY PASS RESTARTS
0008 C30040     00430           JP 4000H        ;RST 1
000B 010003     00440 S1        LD BC,0300H     ;CLEAR 1ST 3 PAGES
000E 1803       00450           JR S2
0010 C30340     00460           JP 4003H        ;RST 2
0013 77         00470 S2        LD (HL),A
0014 ED56       00480           IM 1            ;SET INTERRUPT MODE 1
0016 1803       00490           JR S3
0018 C30640     00500           JP 4006H        ;RST 3
001B EDA1       00510 S3        CPI
001D F3         00520           DI              ;DISABLE INTERRUPTS
001E 1803       00530           JR S4
0020 C30940     00540           JP 4009H        ;RST 4
0023 E22B00     00550 S4        JP PO,CON       ;CONTINUE INITIALIZATION
0026 18EB       00560           JR S2           ;CONTINUE TO CLEAR RAM
0028 C30C40     00570           JP 400CH        ;RST 5
```

```
002B 3EC9       00580 CON      LD A,0C9H
002D 1804       00590          JR CON1
002F 00         00600          NOP              ;FILL
0030 C30F40     00610          JP 400FH         ;RST 6
0033 310043     00620 CON1     LD SP,USERSP     ;SET USER STACK
0036 1814       00630          JR CON2
0038 C31240     00640          JP 4012H         ;RST 7
003B 0D         00650 TABLE    DEFB 0DH
003C 0C         00660          DEFB 0CH
003D 01         00670          DEFB 01H
003E 0B         00680          DEFB 0BH
003F 0A         00690          DEFB 0AH
0040 08         00700          DEFB 08H
0041 09         00710          DEFB 09H
0042 20         00720          DEFB 20H
0043 0D         00730 SIGNON   DEFB 0DH         ;WILL INITIALIZE CURSOR
0044 46         00740          DEFM 'FROLIC:'
004B 01         00750          DEFB 01H         ;END OF MESSAGE
004C 0E09       00760 CON2     LD C,09          ;LOAD RETURNS IN USER RAM
004E 211840     00770          LD HL,TTYP       ;START OF FILL
0051 CDBB03     00780          CALL FILL1
0054 CDFE01     00790          CALL CLEARS      ;CLEAR SCREEN ONLY ON POWER UP
0057 21003C     00800          LD HL,3C00H
005A 223E40     00810          LD (SCREEN),HL
005D ED732440   00820          LD (SSTACK),SP   ;INITIALIZE USER STACK
0061 3EA0       00830          LD A,0A0H        ;CURSOR MARK
0063 324440     00840          LD (CMARK),A     ;PLACE IN RAM
                00850 ;
                00860 ;        RESET START OF PROGRAM
                00870 ;
0066 31BE42     00880 RSTART   LD SP,MONSP-2    ;INITIALIZE STACK
0069 CD0701     00890          CALL CLRBP       ;CLEAR BREAKPOINTS
006C CD8A01     00900 COMAND   CALL PMSG1       ;FROLIC:
006F 218040     00910 COM1     LD HL,BUFFER     ;BUFFER BEGINNING
0072 E5         00920          PUSH HL          ;SAVE FOR LATER
0073 CD2701     00930          CALL BUF         ;BUFFER AVAILABLE FOR OTHERS
0076 362C       00940          LD (HL),2CH      ;DELIMITER
0078 23         00950          INC HL
0079 3601       00960          LD (HL),01H      ;BREAK AT END OF BUFFER
007B E1         00970          POP HL           ;GET BUFFER BEGINNING
007C 77         00980          LD (HL),A        ;PUT 0DH, ERRORS WILL DO LINEFEED
007D E5         00990          PUSH HL
007E FDE1       01000          POP IY           ;BUFFER POINTER FOR MONITOR
0080 31C042     01010 CMND     LD SP,MONSP      ;NECESSARY FOR COMMANDS
0083 210403     01020          LD HL,CONTIN     ;RETURN ADDRESS ON STACK
0086 E5         01030          PUSH HL          ;PUT RETURN ADDRESS ON STACK
0087 CD9E01     01040          CALL NEWLIN      ;SAVES DOING IT IN COMMAND
008A CDEF02     01050 CD       CALL GET         ;GET WILL GET FIRST NON ZERO CHAR
008D 28FB       01060          JR Z,CD          ;IGNORE LEADING SPACES
008F D640       01070          SUB '@'          ;BORROW IF < @
0091 DA0C03     01080          JP C,ERROR       ;A 0C IF `
0094 212D03     01090          LD HL,COMTAB     ;POINT TO COMMAND TABLE
                01100                           ;A BORROW IS ALWAYS GENERATED
0097 FE5A       01110          CP 'Z'           ;BORROW VALID @ THRU Z
0099 D20C03     01120          JP NC,ERROR
009C 87         01130          ADD A,A          ;DOUBLE NUMBER
009D 5F         01140          LD E,A           ;FOR DOUBLE ADD
```

```
009E 1600       01150          LD D,0H          ;BE AT A PAGE BOUNDARY
00A0 19         01160          ADD HL,DE        ;ADD WITH CARRY SET FROM COMPARE
00A1 5E         01170          LD E,(HL)        ;GET LOW BYTE
00A2 23         01180          INC HL
00A3 56         01190          LD D,(HL)        ;GET HIGH BYTE
00A4 EB         01200          EX DE,HL         ;JUMP TO ADDRESS
00A5 E9         01210          JP (HL)          ;EXECUTE REST OF COMMAND
                01220          ;
00A6 E3         01230 RSAVE    EX (SP),HL       ;GET PROGRAM COUNTER
00A7 2B         01240          DEC HL           ;ADJUST
00A8 223C40     01250          LD (PCSAVE),HL   ;AND SAVE
00AB E1         01260          POP HL
00AC 223A40     01270          LD (HLSAVE),HL   ;SAVE HL AND ADJUST STACKPOINTER
00AF F5         01280          PUSH AF          ;SAVE FLAGS IN FORMER HL
00B0 210200     01290          LD HL,0002H      ;ADJUST FOR RST OR INT
00B3 39         01300          ADD HL,SP        ;GET SYSTEM USER STACK
00B4 222440     01310          LD (SSTACK),HL   ;SYSTEM STACK STORAGE
00B7 F1         01320          POP AF           ;GET FLAGS BACK
00B8 313A40     01330          LD SP,HLSAVE     ;START OF REGISTER SAVE
00BB D5         01340          PUSH DE
00BC C5         01350          PUSH BC
00BD F5         01360          PUSH AF          ;FLAGS ON STACK
00BE 3A4240     01370          LD A,(SSFLAG)    ;SINGLE STEP?
00C1 B7         01380          OR A             ;TEST
00C2 280B       01390          JR Z,EXX         ;NO SINGLE STEP IF ZERO
00C4 AF         01400          XOR A            ;NOW ZERO
00C5 324240     01410          LD (SSFLAG),A    ;CORRECT FLAG
00C8 2A3C40     01420          LD HL,(PCSAVE)   ;GET PC COUNTER
00CB 23         01430          INC HL           ;ADJUST
00CC 223C40     01440          LD (PCSAVE),HL   ;REPLACE
00CF D9         01450 EXX      EXX
00D0 E5         01460          PUSH HL
00D1 D5         01470          PUSH DE
00D2 C5         01480          PUSH BC
00D3 08         01490          EX AF,AF'
00D4 F5         01500          PUSH AF
00D5 ED57       01510          LD A,I           ;GET INT VECTOR AND IFF FLAG
00D7 67         01520          LD H,A           ;STORE IN H
00D8 2E00       01530          LD L,00H         ;INITIALIZE L
00DA E2DE00     01540          JP PO,SAVE1      ;IF PARITY FLAG NOT SET IFF INT D
00DD 2C         01550          INC L            ;IFF SET INDICATE WITH NON ZERO
00DE E5         01560 SAVE1    PUSH HL          ;SAVE INT VECTOR AND INT SET FLAG
00DF FDE5       01570          PUSH IY
00E1 DDE5       01580          PUSH IX
00E3 21003C     01590          LD HL,3C00H      ;START OF SCREEN TRANSFERRED
00E6 010004     01600          LD BC,0400H
00E9 ED5B3E40   01610          LD DE,(SCREEN)   ;AREA STARTING AT SCREEN
00ED EDB0       01620          LDIR
00EF FD2A4840   01630          LD IY,(BPOINT)   ;GET LAST POSITION IN BUFFER
00F3 3B         01640          DEC SP           ;MUST NOT DESTROY STACK
00F4 3B         01650          DEC SP
00F5 CD0701     01660          CALL CLRBP       ;CLEAR BREAK POINTS
00F8 31BE42     01670          LD SP,MONSP-2    ;CORRECT STACK
00FB CD8605     01680          CALL XALL        ;DISPLAY REGISTERS
00FE 2A3C40     01690          LD HL,(PCSAVE)   ;GET PROGRAM COUNTER
0101 010500     01700          LD BC,05H        ;NUMBER OF BYTES
0104 C3A703     01710          JP DISP1         ;DISPLAY
```

```
0107 210000    01720 CLRBP   LD HL,00H        ;CLEAR
010A 3A4540    01730         LD A,(BPNO)      ;LD HL WITH NO OF BPOINTS
010D B7        01740         OR A             ;SET ZERO FLAG
010E 6F        01750         LD L,A           ;STORE JUST IN CASE
010F C8        01760         RET Z            ;IF NO BREAKPOINTS GO TO COMMAND
0110 C1        01770         POP BC           ;GET RETURN ADDRESS
0111 29        01780         ADD HL,HL        ;DOUBLE VALUE IN L
0112 29        01790         ADD HL,HL        ;AND AGAIN
0113 EB        01800         EX DE,HL
0114 218040    01810         LD HL,BREAKS
0117 ED52      01820         SBC HL,DE        ;BORROW CLEAR DECREMENT STACK
0119 F9        01830         LD SP,HL         ;NEW STACK POSITION TOP
011A C5        01840         PUSH BC
011B E1        01850         POP HL           ;READY FOR RETURN
011C 47        01860         LD B,A           ;SHOULD STILL HAVE NUMBER
011D F1        01870 CLRBP1  POP AF
011E D1        01880         POP DE
011F 12        01890         LD (DE),A
0120 10FB      01900         DJNZ CLRBP1
0122 AF        01910         XOR A
0123 324540    01920         LD (BPNO),A      ;PUT ZERO IN BPNO
0126 E9        01930         JP (HL)          ;HL HL HAS RETURN ADDRESS
               01940         :
               01950         ;FILL BUFFER WITH COMMAND STRING
               01960         ;
0127 3E40      01970 BUF     LD A,40H         ;IF ENTERED HERE ONLY UPPER CASE
0129 324640    01980         LD (UCFLAG),A    ;40H IS UPPER FLAG, 60H FOR LC
012C 23        01990 BUF0    INC HL           ;LEAVE ROOM FOR BUFFER COUNT
012D 0601      02000         LD B,01H         ;ROOM FOR COUNT.COMMA,AND 01H
012F CD6E02    02010 BUF1    CALL GBYTE       ;GET CHARACTER RETURN WITH DATA
0132 FE01      02020         CP 01H           ;ESCAPE FROM BUFFER ENTRY
0134 CA6600    02030         JP Z,RSTART      ;RETURN TO COMMAND, SP IS OK
0137 FE0D      02040         CP 0DH           ;CARRIAGE RETURN
0139 C8        02050         RET Z            ;RETURN TO CALLING ROUTINE
013A FE0C      02060         CP 0CH           ;CLEAR SCREEN
013C 2812      02070         JR Z,PCH         ;CHANGE DATA TO K COMMAND
013E FE08      02080 DELETE  CP 08H           ;BACKSPACE?
0140 2010      02090         JR NZ,PCHAR      ;NO INSERT CHAR INTO BUFFER
0142 3E01      02100 DLT     LD A,01H         ;BEGINNING
0144 B8        02110         CP B             ;IS IT
0145 28E8      02120         JR Z,BUF1        ;CAN'T GO BACK FURTHER
0147 2B        02130 DLTE    DEC HL           ;DECREMENT BUFFER
0148 05        02140 DLTE1   DEC B            ;AND COUNT
0149 3E08      02150         LD A,08H
014B CDB201    02160         CALL CRT         ;REMOVE FROM SCREEN
014E 18DF      02170         JR BUF1          ;GET NEXT CHAR
0150 3E43      02180 PCH     LD A,'C'         ;CLEAR SCREEN
0152 04        02190 PCHAR   INC B            ;IS THERE ROOM?
0153 28F2      02200         JR Z,DLTE        ;NO SO DO AUTOMATIC 08H
0155 77        02210         LD (HL),A
0156 CDB201    02220         CALL CRT         ;LOAD IN BUFFER AND DISPLAY
0159 23        02230         INC HL           ;NEXT POSITION IN BUFFER
015A 18D3      02240         JR BUF1          ;GET NEXT CHARACTER
               02250         ;
               02260         ;FOLLOWING ARE PRINT ROUTINES AND DATA
               02270         ;CONDITIONING FOR THESE SUBROUTINES
               02280         ;
```

```
015C CDA201    02290 SPCHL   CALL SPACE      ;PRINT A SPACE FOLLOWED BY
015F 7E        02300 PPHL    LD A,(HL)       ;THE  NUMBER LOCATED AT (HL)
0160 CD6301    02310 PHEX    CALL PHEXH      ;TWO NIBBLES MUST BE OUTPUT
0163 0F        02320 PHEXH   RRCA            ;GET HIGH HEX VALUE
0164 0F        02330         RRCA
0165 0F        02340         RRCA
0166 0F        02350         RRCA            ;SWAP COMPLETE
0167 F5        02360 PHEXL   PUSH AF         ;GET LOW HEX VALUE
0168 E60F      02370         AND 0FH         ;MASK
016A FE0A      02380         CP 0AH          ;GREATER THAN 9
016C 3007      02390         JR NC,ADJUST    ;THEN NO CARRY
016E F630      02400         OR 30H          ;NOW ASCII FOR 0-9
0170 CDA401    02410 PHEX1   CALL PRNT       ;OUTPUT
0173 F1        02420         POP AF          ;GET AND ADJUST SP
0174 C9        02430         RET
0175 C637      02440 ADJUST  ADD A,37H       ;ADJUST FOR ALPHABETIC
0177 18F7      02450         JR PHEX1
               02460         ;
               02470         ;PRINTS NUMBER IN HL REGISTER AS ADDRESS
               02480         ;
0179 7C        02490 PNHL    LD A,H          ;PRINT H FIRST
017A CD6001    02500         CALL PHEX
017D 7D        02510         LD A,L          ;NOW  L
017E 18E0      02520         JR PHEX         ;RETURNS FROM CRT PROGRAM
0180 CD9E01    02530 ADDR    CALL NEWLIN     ;DO CR AND LINE FEED
0183 CD7901    02540 ADDR1   CALL PNHL       ;PRINTS HL WHICH IS ADDRESS
0186 3E3A      02550 ADDR2   LD A,':'        ;DELIMITER
0188 181A      02560         JR PRNT
               02570         ;
               02580         ;PRINT MESSAGE
               02590         ;
018A 214300    02600 PMSG1   LD HL,SIGNON    ;FROLIC:
018D 7E        02610 PMSG    LD A,(HL)       ;GET DATA
018E FE01      02620         CP 01H          ;END OF MESSAGE MARK
0190 C8        02630         RET Z           ;RETURNS FROM MESSAGES VIA BREAK
0191 CDB201    02640         CALL CRT
0194 23        02650         INC HL          ;NEXT DATA
0195 18F6      02660         JR PMSG
               02670         ;
               02680         ;SHIFT DATA THROUGH HL PAIR
               02690         ;
0197 29        02700 LDHL    ADD HL,HL       ;SHIFT
0198 29        02710         ADD HL,HL
0199 29        02720         ADD HL,HL
019A 29        02730         ADD HL,HL
019B B5        02740         OR L            ;OR AC IN SHIFT PATTERN
019C 6F        02750         LD L,A          ;REPLACE L
019D C9        02760         RET
               02770         ;
               02780         ;PRINTS DATA IN A TO PRINT DEVICE
               02790         ;
019E 3E0D      02800 NEWLIN  LD A,0DH        ;LINE FEED
01A0 1802      02810         JR PRNT         ;WANT TO BREAK ON NEW LINES
01A2 3E20      02820 SPACE   LD A,20H        ;SPACE ADJUST
01A4 F5        02830 PRNT    PUSH AF         ;SAVE
01A5 CD7E02    02840 PRNT1   CALL CHKIN      ;HAS BREAK OR SPACE
01A8 FE01      02850         CP 01H          ;BEEN PRESSED
```

```
01AA CA6600    02860          JP Z,RSTART      ;CONTINUE WITH BUFFER
01AD FE20      02870          CP 20H
01AF 28F4      02880          JR Z,PRNT1       ;YES, WAIT FOR RELEASE
01B1 F1        02890          POP AF           ;FALL INTO CRT DISPLAY
               02900          ;
               02910          ;CRT DISPLAY
               02920          ;
01B2 D9        02930 CRT      EXX              ;CRT OUTPUT OF DATA IN A
01B3 F5        02940          PUSH AF
01B4 DBFF      02950          IN A,(0FFH)      ;CHECK FOR LARGE GRAPHICS
01B6 E640      02960          AND 40H          ;ZERO LARGE,NZ REGULAR
01B8 0600      02970          LD B,00H         ;FLAG FOR LARGE CHARACTERS
01BA 2001      02980          JR NZ,CRT1
01BC 04        02990          INC B            ;FLAG FOR REGULAR CHARACTERS
01BD F1        03000 CRT1     POP AF           ;GET CHARACTER BACK
01BE F5        03010          PUSH AF          ;REPLACE ON STACK
01BF 2A4040    03020          LD HL,(CURSOR)
01C2 3620      03030          LD (HL),20H      ;REMOVE CURSOR
01C4 FE0D      03040          CP 0DH           ;CARRIAGE RETURN
01C6 280D      03050          JR Z,LINEF
01C8 FE08      03060          CP 08H           ;BACKSPACE
01CA 281B      03070          JR Z,BKSP
01CC 77        03080          LD (HL),A        ;PLACE CHARACTER ON SCREEN
01CD 23        03090          INC HL           ;ADVANCE CURSOR
01CE 1001      03100          DJNZ FULSCN
01D0 23        03110          INC HL
01D1 CB74      03120 FULSCN   BIT 6,H          ;OFF PAGE?
01D3 2816      03130          JR Z,OUT2
01D5 01C003    03140 LINEF    LD BC,03C0H      ;1024-64 MEMORY BYTES
01D8 11003C    03150          LD DE,VIDEO
01DB 21403C    03160          LD HL,VIDEO+64D  ;ONE LINE FROM PAGE TOP
01DE EDB0      03170          LDIR
01E0 EB        03180          EX DE,HL         ;CLEAR LAST LINE
01E1 CD0302    03190          CALL CLEAR       ;CLEAR LAST LINE
01E4 21C13F    03200          LD HL,VIDEO+961D        ;BEGIN OF LAST LINE
01E7 2B        03210 BKSP     DEC HL
01E8 1001      03220          DJNZ OUT2
01EA 2B        03230          DEC HL
01EB 224040    03240 OUT2     LD (CURSOR),HL
01EE 3A4440    03250          LD A,(CMARK)     ;MONITOR OR USER?
01F1 77        03260          LD (HL),A        ;CURSOR MARK
01F2 F1        03270          POP AF           ;GET DATA FROM STACK
01F3 4F        03280          LD C,A           ;SAVE IN C REGISTER
01F4 3A4A40    03290          LD A,(PRNTFG)    ;GET HARD COPY FLAG
01F7 B7        03300          OR A             ;IF NON ZERO CALL PRINT
01F8 C41840    03310          CALL NZ,TTYP     ;IN RAM A JUMP TO USER ROUTINE
01FB 79        03320          LD A,C           ;RESTORE DATA TO A
01FC D9        03330          EXX              ;REPLACE THOSE REGISTERS
01FD C9        03340          RET
01FE 21003C    03350 CLEARS   LD HL,VIDEO
0201 FD23      03360          INC IY           ;CORRECT BUFFER
0203 CB74      03370 CLEAR    BIT 6,H          ;FINISHED
0205 C0        03380          RET NZ           ;WHEN THIS BIT GOES ONE
0206 3620      03390          LD (HL),20H      ;SPACE ON SCREEN
0208 23        03400          INC HL           ;NEXT
0209 18F8      03410          JR CLEAR         ;CONTINUE WITH CLEAR
               03420          ;
```

```
                03430           ;FILL ROUTINES FOR ADDRESS OR DATA
                03440           ;
020B CD1402     03450 G3N       CALL G2N        ;GET TWO NUMBERS HL AND BC
020E EB         03460           EX DE,HL        ;SAVE HL VALUE IN DE
020F CD4302     03470           CALL GHL        ;GET 3RD ARGUMENT
0212 EB         03480           EX DE,HL        ;THIS GOES IN DE REGISTER
0213 C9         03490           RET             ;WITH HL START ADDRESS
0214 CD4302     03500 G2N       CALL GHL        ;GET 2 NUMBERS FIRST FOR HL
0217 EB         03510           EX DE,HL
0218 CDF102     03520           CALL GET1       ;GET LAST DATA IN BUFFER
021B 3821       03530           JR C,G2N3       ;FOR LONGEST INSTRUCTION
021D CCEF02     03540 G2N0      CALL Z,GET      ;EITHER COULD BE T
0220 28FB       03550           JR Z,G2N0       ;SKIP SPACES
0222 FE54       03560           CP 'T'          ;THROUGH TILL LAST ADDRESS
0224 280D       03570           JR Z,G2N1       ;GO COMPUTE BLOCK LENGTH
0226 FD2B       03580           DEC IY          ;ADJUST POINTER
0228 CDF102     03590           CALL GET1       ;LAST IN BUFFER
022B C20C03     03600           JP NZ,ERROR     ;ONLY VALID DELIMITERS
022E CD4302     03610           CALL GHL        ;GET BLOCK LENGTH
0231 1807       03620           JR G2N2         ;GO STORE IN HL
0233 CD4302     03630 G2N1      CALL GHL        ;END ADDRESS MUST COMPUTE
0236 3F         03640           CCF             ;CLEAR FLAG
0237 ED52       03650           SBC HL,DE
0239 23         03660           INC HL          ;NEED ONE MORE
023A 44         03670 G2N2      LD B,H          ;STORE LENGTH IN BC
023B 4D         03680           LD C,L
023C EB         03690           EX DE,HL        ;PUT START ADDRESS BACK
023D C9         03700           RET
023E 214000     03710 G2N3      LD HL,40H       ;MINIMUM LENGTH 40H
0241 18F7       03720           JR G2N2         ;STORE IN BC
                03730           ;
                03740           ;GET VALID HEX IN HL REGISTER
                03750           ;
0243 210000     03760 GHL       LD HL,0000H     ;INITIALIZE TO ZERO
0246 CDEF02     03770 GHL0      CALL GET
0249 28FB       03780           JR Z,GHL0       ;SKIP SPACES
024B CD5D02     03790           CALL VALHEX     ;CHECK FOR VALIDITY
024E DA0C03     03800           JP C,ERROR      ;THIS MUST CLEAR BREAK POINTS
0251 CD9701     03810 GHL1      CALL LDHL       ;SHIFT INTO HL PAIR
0254 CDEF02     03820           CALL GET
0257 CD5D02     03830           CALL VALHEX
025A 30F5       03840           JR NC,GHL1
025C C9         03850           RET             ;AND RETURN TO CALLING PROGRAM
                03860           ;
                03870           ;CHECK IF VALID HEX NO.
                03880           ;
025D C6D0       03890 VALHEX    ADD A,0D0H      ;IF BETWEEN 30 AND 39
025F 3F         03900           CCF             ;AND 41 AND 47 CARRY GENERATED
0260 D8         03910           RET C           ;RETURN NOT VALID FLAG
0261 FE0A       03920           CP 0AH          ;IF LESS THAN 10 VALID
0263 3F         03930           CCF             ;VALID RETURN IS NO CARRY
0264 D0         03940           RET NC          ;BETWEEN 0 AND 9 AND VALID
0265 C6F9       03950           ADD A,0F9H      ;IF BETWEEN 17H AND 22H
0267 FE0A       03960           CP 0AH          ;THIS WILL FLAG
0269 D8         03970           RET C           ;ERROR THOSE FEW
026A FE10       03980           CP 10H
026C 3F         03990           CCF             ;IF OVER 0FH NO CARRY
```

```
026D C9          04000          RET              ;RETURN WITH PROPER FLAG
                 04010          ;
                 04020          ;GET CHARACTERS
                 04030          ;
026E E5          04040 GBYTE    PUSH HL
026F CD7E02      04050          CALL CHKIN
0272 214740      04060          LD HL,OLD
0275 BE          04070          CP (HL)
0276 28F7        04080          JR Z,GBYTE+1     ;WANT NEW CHARACTER
0278 77          04090          LD (HL),A
0279 B7          04100          OR A             ;NO NULLS
027A 28F3        04110          JR Z.GBYTE+1
027C E1          04120          POP HL
027D C9          04130          RET
027E 3A7F38      04140 CHKIN    LD A,(387FH)
0281 B7          04150          OR A
0282 C8          04160          RET Z
0283 D9          04170          EXX
0284 CD9402      04180 TWO      CALL KYD1        ;GET CHARACTER
0287 57          04190          LD D,A           ;SAVE
0288 0600        04200          LD B,00H         ;SHORT DELAY
028A 10FE        04210 D1       DJNZ D1
028C CD9402      04220          CALL KYD1
028F BA          04230          CP D             ;IS IT THE SAME
0290 20F2        04240          JR NZ,TWO        ;NOT SAME TRY AGAIN
0292 D9          04250          EXX
0293 C9          04260          RET
                 04270          ;
                 04280          ;KEY SCAN
                 04290          ;
0294 010000      04300 KYD1     LD BC,0000H
0297 210138      04310          LD HL,3801H      ;TURN ON BIT ZERO SCAN TO START
029A 7E          04320 NEXT     LD A,(HL)        ;GET DATA
029B B7          04330          OR A
029C 2008        04340          JR NZ,HDATA
029E CB25        04350          SLA L
02A0 03          04360          INC BC
02A1 F29A02      04370          JP P,NEXT
02A4 AF          04380          XOR A
02A5 C9          04390          RET              ;ESCAPE
02A6 1F          04400 HDATA    RRA
02A7 3803        04410          JR C,LOWBIT
02A9 04          04420          INC B
02AA 18FA        04430          JR HDATA
02AC CB75        04440 LOWBIT   BIT 6,L          ;LINE 6 SPECIAL CHARACTERS
02AE 2037        04450          JR NZ,SPEC
02B0 79          04460          LD A,C
02B1 07          04470          RLCA
02B2 07          04480          RLCA
02B3 07          04490          RLCA
02B4 B0          04500          OR B
02B5 4F          04510          LD C,A           ;5 LSB HAVE LOW DATA
02B6 3A8038      04520          LD A,(3880H)     ;IS SHIFT KEY PRESSED?
02B9 47          04530          LD B,A           ;THIS INFORMATION IS IN B
02BA 3E08        04540          LD A,08H
02BC BD          04550          CP L             ;IF CARRY NOT ALPHA CHAR
02BD 380C        04560          JR C,NALPHA
```

```
02BF 3A4640    04570          LD A,(UCFLAG)    ;60H HERE IF LOWERCASE REQUIRED
               04580                           ;OTHERWISE 40H FOR ONLY UPPERCASE
02C2 B1        04590          OR  C
02C3 CB40      04600          BIT 0,B          ;WAS SHIFT PRESSED
02C5 2001      04610          JR NZ,CAPIT
02C7 C9        04620          RET
02C8 CBAF      04630 CAPIT    RES 5,A          ;AC HAS UPPER CASE DATA
02CA C9        04640          RET
02CB CB6D      04650 NALPHA   BIT 5,L
02CD 200B      04660          JR NZ,NOT7
02CF 3E30      04670 NALPH1   LD A,30H
02D1 B1        04680          OR C
02D2 CB40      04690          BIT 0,B
02D4 2001      04700          JR NZ.CHAR       ;THIS IS NOW CHARACTER
02D6 C9        04710          RET
02D7 CBA7      04720 CHAR     RES 4,A
02D9 C9        04730          RET
02DA 3A2038    04740 NOT7     LD A,(3820H)
02DD FE10      04750          CP 10H
02DF 38EE      04760          JR C,NALPH1
02E1 78        04770          LD A,B
02E2 EE01      04780          XOR 01H
02E4 47        04790          LD B,A
02E5 18E8      04800          JR NALPH1
02E7 213B00    04810 SPEC     LD HL,TABLE      ;TABLE MUST BE LOCATED
02EA 78        04820          LD A,B           ;SUCH THAT THE FOLLOWING ADDITION
02EB 85        04830          ADD A,L          ;DOES NOT CROSS BOUNDRIES
02EC 6F        04840          LD L,A
02ED 7E        04850          LD A,(HL)
02EE C9        04860          RET
               04870          ;
               04880          ;GET FETCHES NEXT DATA IN INPUT STRING
               04890          ;BY INC FIRST IY WILL ALWAYS POINT TO LAST VALUE
               04900          ;
02EF FD23      04910 GET      INC IY
02F1 FD7E00    04920 GET1     LD A,(IY+0H)     ;ENTER HERE FOR LAST ENTRY
02F4 FE01      04930          CP 01H           ;BREAK?
02F6 CA6C00    04940          JP Z,COMAND      ;START ALL OVER
02F9 FE20      04950          CP 20H           ;SPACE SET ZERO FLAG
02FB C8        04960          RET Z            ;FLAG SPACES
02FC FE2C      04970          CP ','           ;IF COMMA SET CARRY
02FE C0        04980          RET NZ           ;ALL OTHERS RETURN
02FF B7        04990 GET3     OR A             ;NO ZERO
0300 37        05000          SCF              ;SET CARRY
0301 D8        05010          RET C
               05020          ;
0302 FD23      05030 CONTI1   INC IY           ;NEXT
0304 CDF102    05040 CONTIN   CALL GET1        ;LAST CHARACTER IN STRING
0307 28F9      05050          JR Z,CONTI1      ;SKIP TRAILING SPACES
0309 DA8000    05060          JP C,CMND        ;FINISH OLD OR BACK TO BEGIN
030C 3E3F      05070 ERROR    LD A,'?'         ;SHOW AT END
030E FD7700    05080          LD (IY+0),A      ;SHOW IT
0311 CDEF02    05090 ER0      CALL GET         ;ADJUST BUFFER
0314 3802      05100          JR C,ER1         ;WANT NEXT COMMA
0316 18F9      05110          JR ER0           ;OUT ONLY WITH TRUE COMMA
0318 218040    05120 ER1      LD HL,BUFFER     ;0DH STARTS THIS LINE
031B CD8D01    05130          CALL PMSG        ;OUTPUT BUFFER TEXT TO ERROR
```

```
031E CD0701    05140           CALL CLRBP       ;CLEAR BREAKPOINTS
0321 DBFF      05150 TOFF1     IN A,(0FFH)      ;GET STATUS OF SCREEN
0323 E640      05160           AND 40H          ;IF ONE SMALL GRAPHIC
0325 2002      05170           JR NZ,ER2        ;TURN OFF GRAPHIC
0327 3E08      05180           LD A,08H         ;LD GRAPHIC CONTROL
0329 D3FF      05190 ER2       OUT (0FFH),A     ;ALL SET
032B 18D7      05200           JR CONTIN        ;CONTINUE WITH BUFFER
               05210           ;
               05220           ;COMMAND TABLE
               05230           ;
032D B703      05240 COMTAB    DEFW FILL        ;FILL ADDRESS SPACE WITH CONSTANT
032F 6303      05250           DEFW ASCII       ;ADDRESS OF ROUTINE
0331 7A03      05260           DEFW BUFFR       ;ADDRESS OF BUFFER PROGRAM
0333 FE01      05270           DEFW CLEARS      ;CLEAR SCREEN WITH 20H
0335 A403      05280           DEFW DISPLY      ;ADDRESS OF ROUTINE
0337 D903      05290           DEFW GO          ;EXECUTE WITH CURRENT SCREEN
0339 D807      05300           DEFW FINDON      ;SEARCH FOR SINGLE BYTE
033B C603      05310           DEFW ENTER       ;EXECUTE WITH OLD SCREEN DATA
033D 7804      05320           DEFW HEXM        ;ADDRESS OF HEXM
033F CF04      05330           DEFW SUBSM       ;INSERT DATA FROM BUFFER
0341 0C03      05340           DEFW ERROR       ;J
0343 0C03      05350           DEFW ERROR       ;K
0345 AC04      05360           DEFW LIST        ;TOGGLE HARDCOPY SWITCH
0347 B704      05370           DEFW MOVE        ;ADDRESS OF MOVE
0349 8D04      05380           DEFW IN          ;INPUT FROM PORT
034B C504      05390           DEFW OUT         ;OUTPUT TO PORT
034D 1B40      05400           DEFW PROM        ;PROM PROGRAMMER
034F D407      05410           DEFW FINDTO      ;SEARCH FOR TWO BYTE INTEL FORMAT
0351 2307      05420           DEFW READ        ;READ A CASSETTE TAPE WITH BIAS
0353 F204      05430           DEFW INSERT      ;DATA FROM KEYBOARD
0355 C603      05440           DEFW ENTER       ;TRANSFER FROM SCREEN TO VIDEO
0357 1E40      05450           DEFW USER        ;USER ROUTINE
0359 3F05      05460           DEFW VERIFY      ;ADDPESS OF VERIFY
035B 6406      05470           DEFW WRIT        ;WRITE A TAPE ON CASSETTE
035D 5E05      05480           DEFW EXAM        ;ADDRESS
035F 0C03      05490           DEFW ERROR
0361 C103      05500           DEFW STEP        ;SINGLE STEP IF HARDWARE ADDED
               05510                            ;WITH OLD SCREEN DATA
               05520           ;
               05530           ;COMMAND SUBROUTINES
               05540           ;
               05550           ;DISPLAY IN ASCII FORMAT WITH GRAPHICS
               05560           ;
0363 CD1402    05570 ASCII     CALL G2N         ;GET ARGUMENTS
0366 1620      05580 ASCII1    LD D,32D         ;56 DECIMAL POSITIONS MAX
0368 CD8001    05590           CALL ADDR        ;PRINT LOCATION OF BEGINNING LINE
036B CDA201    05600           CALL SPACE       ;SPACE
036E 7E        05610 ASCII2    LD A,(HL)        ;GET DATA
036F CDB201    05620 ASCII4    CALL CRT
0372 EDA1      05630           CPI              ;ASCII DUMP FINISHED?
0374 E0        05640           RET PO           ;CHANGE LATER FOR STRING
0375 15        05650           DEC D            ;FINISHED WITH LINE?
0376 28EE      05660           JR Z,ASCII1      ;NOT FINISHED WITH LINE
0378 18F4      05670           JR ASCII2        ;NOT FINISHED
               05680           ;
               05690           ;EUFFER SAVE ANB EXECUTE ROUTINES
               05700           ;
```

```
037A CDEF02    05710 BUFFR    CALL GET          ;GET NEXT DATA
037D DA9303    05720          JP C,TRAN         ;TRANSFER SAVED BUFFER
0380 FE53      05730          CP 'S'            ;SAVE BUFFER
0382 2801      05740          JR Z,SAVE         ;IGNORE ALL ELSE
0384 C9        05750          RET               ;CONTINUE AND IGNORE
0385 218040    05760 SAVE     LD HL,BUFFER      ;BUFFER BEGINNING
0388 010001    05770          LD BC,100H        ;PLENTY
038B 118041    05780          LD DE,TEM         ;PLACE TO SAVE
038E EDB0      05790          LDIR              ;TRANSFER COMMAND TEXT
0390 FD23      05800          INC IY            ;BYPASS THE S FOR CONTIN
0392 C9        05810          RET
0393 218041    05820 TRAN     LD HL,TEM
0396 010001    05830          LD BC,100H        ;ONE PAGE OF BUFFER
0399 118040    05840          LD DE,BUFFER
039C D5        05850          PUSH DE
039D FDE1      05860          POP IY
039F EDB0      05870          LDIR
03A1 C38A00    05880          JP CD             ;QUICK WAY TO LOAD BUFFER
               05890          ;
               05900          ;DISPLAY IN HEXFORMAT
               05910          ;
03A4 CD1402    05920 DISPLY   CALL G2N          ;GET ARGUMENTS
03A7 CD8001    05930 DISP1    CALL ADDR         ;PRINT MEM ADDRESS
03AA CD5C01    05940 DISP2    CALL SPCHL        ;SPACE FOLLOWED BY HEX DATA
03AD EDA1      05950          CPI               ;INCREMENT POINTERS
03AF E0        05960          RET PO            ;EC=0 THEN DONE
03B0 7D        05970          LD A,L            ;GET ADDRESS OF MEM
03B1 E60F      05980          AND 0FH           ;MASK ALL BUT LOW
03B3 28F2      05990          JR Z,DISP1        ;NEXT LINE
03B5 18F3      06000          JR DISP2          ;GO FINISH LINE
               06010          ;
               06020          ;
               06030          ;
               06040          ;FILL MEMORY WITH CONSTANT
               06050          ;
03B7 CD0B02    06060 FILL     CALL G3N          ;GET 3ARGUMENTS, WHERE,
03BA 7B        06070          LD A,E            ;HOW MANY AND WHAT
03BB 77        06080 FILL1    LD (HL),A         ;DO IT
03BC EDA1      06090          CPI               ;INCREMENT POINTERS
03BE E0        06100          RET PO            ;FINISHED?
03BF 18FA      06110          JR FILL1          ;CONTINUE
               06120          ;
               06130          ;SINGLE STEP EXECUTE WITH LAST SCREEN DATA
               06140          ;
03C1 3E01      06150 STEP     LD A,01           ;SINGLE STEP FLAG
03C3 324240    06160          LD (SSFLAG),A     ;STORE AND FALL INTO GO
               06170          ;
03C6 2A3E40    06180 ENTER    LD HL,(SCREEN)    ;CURRENT SCREEN STORAGE
03C9 11003C    06190          LD DE,3C00H       ;TO BE PLACED HERE
03CC 010004    06200          LD BC,0400H       ;ALL 1024 OF THEM
03CF EDB0      06210          LDIR              ;DOES IT
03D1 CDF102    06220          CALL GET1         ;LAST IN STRING
03D4 FE54      06230          CP 'T'            ;TRANSFER IS ALL THAT IS
03D6 CA6F00    06240          JP Z,COM1         ;REQUIRED THEN RETURN NO PROMPT
               06250          ;
               06260          ;GET PC AND EXECUTE WITHOUT SCREEN
               06270          ;IF ENTERED AT GO
```

```
                06280           ;
03D9 0600       06290 GO        LD B,00H            ;BREAK COUNTER
03DB FD23       06300           INC IY              ;ADJUST BUFFER
03DD 211240     06310           LD HL,RSTLOC
03E0 36C3       06320           LD (HL),0C3H        ;PUT JUMP THERE
03E2 21A600     06330           LD HL,RSAVE         ;ADDRESS FOR JUMP
03E5 221340     06340           LD (RSTLOC+1),HL;ADDRESS
03E8 318040     06350           LD SP,BREAKS        ;BREAKSTACK
03EB CDF102     06360 GO1       CALL GET1           ;NEXT DATA IN BUFFER
03EE CCEF02     06370 GO2       CALL Z,GET          ;NEXT IN BUFFER
03F1 28FB       06380           JR Z,GO2
03F3 3841       06390           JR C,LDREG          ;USE STORED PC
03F5 FE2E       06400           CP '.'              ;BREAKPOINT?
03F7 280A       06410           JR Z,BREAK          ;IT IS A BREAK
03F9 FD2B       06420           DEC IY              ;ADJUST FOR 1ST DIGIT OF ADDR
03FB CD4302     06430           CALL GHL            ;START OR BREAK
03FE 223C40     06440           LD (PCSAVE),HL      ;NEW PC
0401 18E8       06450           JR GO1              ;CONTINUE
0403 CDEF02     06460 BREAK     CALL GET            ;NEXT CHARACTER
0406 FE2B       06470           CP '+'              ;RELATIVE JUMP
0408 2816       06480           JR Z,PLUS
040A FE2D       06490           CP '-'
040C 281C       06500           JR Z,MINUS
040E FD2B       06510           DEC IY              ;NO RELATIVE JUMP
0410 CD4302     06520           CALL GHL            ;GET ABSOLUTE VALUE
0413 7E         06530 BREAK1    LD A,(HL)           ;GET CODE AT ADDRESS
0414 E5         06540           PUSH HL             ;SAVE ADDRESS
0415 F5         06550           PUSH AF             ;SAVE DATA
0416 3EFF       06560           LD A,RST            ;RESTART
0418 77         06570           LD (HL),A           ;SAVE IT AS BREAK
0419 04         06580           INC B               ;BREAK COUNTER
041A 78         06590           LD A,B
041B 324540     06600           LD (BPNO),A         ;SAVE
041E 18CB       06610           JR GO1              ;CONTINUE
0420 CD4302     06620 PLUS      CALL GHL            ;GET RELATIVE VALUE
0423 EB         06630           EX DE,HL            ;SAVE IN DE
0424 2A3C40     06640           LD HL,(PCSAVE)      ;PROGRAM COUNTER
0427 19         06650           ADD HL,DE           ;BREAKPOINT COMPUTED
0428 18E9       06660           JR BREAK1
042A CD4302     06670 MINUS     CALL GHL            ;GET RELATIVE VALUE
042D EB         06680           EX DE,HL
042E 2A3C40     06690           LD HL,(PCSAVE)
0431 B7         06700           OR A                ;CLEAR CARRY
0432 ED52       06710           SBC HL,DE
0434 18DD       06720           JR BREAK1           ;RELATIVE JUMP COMPUTED
0436 FD224840   06730 LDREG     LD (BPOINT),IY      ;SAVE BUFFER POINTER AT BPOINT
043A 21D300     06740           LD HL,00D3H         ;OUT INSTRUCTION PORT 00
043D 22A042     06750           LD (JMP),HL
0440 21FBC3     06760           LD HL,0C3FBH        ;ENABLE INT AND JUMP
0443 22A242     06770           LD (JMP+2),HL
0446 2A3C40     06780           LD HL,(PCSAVE)      ;EXECUTE ADDRESS
0449 22A442     06790           LD (JMP+4),HL       ;ADDRESS IN PLACE
044C 312640     06800           LD SP,BOTMS         ;REGISTER DATA
044F DDE1       06810           POP IX
0451 FDE1       06820           POP IY              ;RESTORE IX AND IY
0453 F1         06830           POP AF              ;INTERRUPT INFORMATION
0454 ED47       06840           LD I,A              ;LOAD INTERRUPT VECTOR
```

```
0456 3001      06850          JR NC,GO4        ;IF CARRY NO INTERRUPTS
0458 FB        06860          EI               ;INTERRUPTS REQUESTED
0459 F1        06870 GO4      POP AF           ;THIS TIME IT IS AF'
045A C1        06880          POP BC
045B D1        06890          POP DE
045C E1        06900          POP HL           ;PRIMES IN PLACE
045D D9        06910          EXX
045E 08        06920          EX AF,AF'        ;NOW SWITCH
045F F1        06930          POP AF
0460 C1        06940          POP BC
0461 D1        06950          POP DE
0462 2A2440    06960          LD HL,(SSTACK)   ;GET SYSTEM STACK POINTER
0465 F9        06970          LD SP,HL         ;STACK IN PLACE
0466 2A3A40    06980          LD HL,(HLSAVE)   ;HL OK
0469 F5        06990          PUSH AF          ;SAVE AF ON USER STACK
046A 3A4240    07000          LD A,(SSFLAG)    ;SINGLE STEP?
046D B7        07010          OR A             ;THIS IS HOW WE FIND OUT
046E 2804      07020          JR Z,POPAF
               07030          ;
               07040          ;WITH HARDWARE TO PRODUCE INTERRUPT
               07050          ;
0470 F1        07060          POP AF
0471 C3A042    07070          JP JMP           ;DO OUTPUT AND ENABLE INT
0474 F1        07080 POPAF    POP AF           ;RESTORE REGISTERS
0475 C3A342    07090          JP JMP+3         ;GO EXECUTE
               07100          ;
               07110          ;HEX MATH FIRST SUM THEN DIFFERENCE
               07120          ;
0478 CD4302    07130 HEXM     CALL GHL         ;GET ARGUMENTS
047B EB        07140          EX DE,HL
047C CD4302    07150          CALL GHL         ;2ND ARGUMENT
047F E5        07160          PUSH HL          ;SAVE 2ND ARGUMENT
0480 19        07170          ADD HL,DE        ;GET SUM
0481 CD8301    07180          CALL ADDR1       ;NUMBER FOLLOWED BY :
0484 E1        07190          POP HL           ;GET 2ND BACK
0485 EB        07200          EX DE,HL         ;PUT IN PROPER ORDER
0486 AF        07210          XOR A            ;CLEAR BORROW FLAG
0487 ED52      07220          SBC HL,DE        ;GET DIFFERENCE
0489 CD7901    07230          CALL PNHL        ;DISPLAY SECOND NUMBER
048C C9        07240          RET              ;FINISHED
               07250          ;
               07260          ;INPUT FROM PORT XX AND DISPLAY
               07270          ;
048D CD1402    07280 IN       CALL G2N         ;GET PORT DESIRED
0490 41        07290          LD B,C           ;NUMBER OF INPUT PORTS IN SEQUENCE
0491 4D        07300          LD C,L           ;C HAS PORT TO INPUT
0492 79        07310 INNEXT   LD A,C           ;A GETS PORT NUMBER
0493 CD6001    07320          CALL PHEX        ;OUTPUT PORT NUMBER
0496 CD8601    07330          CALL ADDR2       ;OUTPUT A :
0499 CDA201    07340          CALL SPACE
049C ED78      07350          IN A,(C)         ;GET DATA
049E CD6001    07360          CALL PHEX        ;OUTPUT DATA
04A1 CDA201    07370          CALL SPACE       ;SPACE
04A4 CDA201    07380          CALL SPACE
04A7 05        07390          DEC B            ;ALL PORTS INPUT
04A8 C8        07400          RET Z            ;YES, COMMAND STRING
04A9 0C        07410          INC C            ;NEXT PORT
```

```
04AA 18E6      07420          JR INNEXT
               07430          ;
               07440          ;TURN ON HARD COPY FLAG
               07450          ;
               07460          ;
04AC FD23      07470 LIST     INC IY           ;ADJUST POINTER
04AE 3A4A40    07480          LD A,(PRNTFG)    ;GET CURRENT FLAG
04B1 EE01      07490          XOR 01H          ;TOGGLE TO OPPOSITE STATE
04B3 324A40    07500          LD (PRNTFG),A    ;RETURN TO STORAGE
04B6 C9        07510          RET              ;CONTINUE STRING
               07520          ;
               07530          ;MOVE MEMORY
               07540          ;
04B7 CD0B02    07550 MOVE     CALL G3N         :GET ARGUMENTS OF FROM
04BA E5        07560 MOV1     PUSH HL          ;HOW MUCH AND TO
04BB D5        07570          PUSH DE
04BC C5        07580          PUSH BC          ;SAVE ARGUMENTS FOR VERIFY
04BD EDB0      07590          LDIR             ;ONE OF THOSE NICE Z-80 CODES
04BF C1        07600          POP BC           ;RESTORE
04C0 D1        07610          POP DE
04C1 E1        07620          POP HL           ;RESTORED NOW
04C2 C34205    07630          JP VERIF1        ;JUMP TO VERIFY
               07640          ;
               07650          ;OUTPUT HEX DATA TO PORT XX
               07660          ;
04C5 CD4302    07670 OUT      CALL GHL         ;GET PORT NUMBER
04C8 4D        07680          LD C,L           ;PORT MUST BE IN C
04C9 CD4302    07690          CALL GHL         ;GET DATA
04CC ED69      07700          OUT (C),L        ;OUTPUT
04CE C9        07710          RET              ;FINISHED
               07720          ;
               07730          ;SUBSTITUTE MEMORY FROM STRING OR KEYBOARD
               07740          ;
04CF CD4302    07750 SUBSM    CALL GHL         ;GET ADDRESS
04D2 CD8001    07760 SUBS     CALL ADDR        ;PRINT IT
04D5 E5        07770 SUBS1    PUSH HL          ;SAVE IT
04D6 CD4302    07780          CALL GHL         ;GET NUMBER IN HL
04D9 CDF102    07790          CALL GET1        ;LAST IN STRING
04DC FE2F      07800          CP '/'           ;ADDRESS OR DATA
04DE 280D      07810          JR Z,NEWAD1      ;IT IS ADJUST STACK AND IY
04E0 7D        07820 SUBS3    LD A,L           ;ONLY L USED
04E1 E1        07830          POP HL           ;GET ADDRESS
04E2 77        07840          LD (HL),A        ;STORE
04E3 CD5C01    07850          CALL SPCHL       ;SHOW US
04E6 23        07860          INC HL           ;NEXT FORWARD POSITION
04E7 CDF102    07870          CALL GET1        ;LAST STRING ENTRY
04EA D8        07880          RET C            ;WAY OUT
04EB 18E8      07890          JR SUBS1         ;CONTINUE INSERTION OF DATA
04ED FD23      07900 NEWAD1   INC IY           ;ADJUST IY
04EF D1        07910 NEWADR   POP DE           ;DON'T DESTROY NEW ADDRESS
04F0 18E0      07920          JR SUBS          ;PRINT NEW ADDRESS
               07930          ;
               07940          ;FROM KEYBOARD
               07950          ;
04F2 CD4302    07960 INSERT   CALL GHL         ;GET ADDRESS
04F5 CD8001    07970 INS1     CALL ADDR
04F8 CD5C01    07980 INS2     CALL SPCHL       ;SPACE FOLLOWED BY DATA
```

```
04FB E5       07990          PUSH HL          ;SAVE ADDRESS
04FC 210000   08000          LD HL,00H        ;CLEAR DATA
04FF 0601     08010          LD B,01H         ;COUNT
0501 3E3F     08020 PRINIT   LD A,'?'         ;PROMPT FOR SUBSTITUTE KEYBOARD
0503 CDB201   08030          CALL CRT         ;TELL WE ARE LOOKING FOR DATA
0506 CD6E02   08040 INS3     CALL GBYTE       ;DATA FROM KEYBOARD
0509 FE0D     08050          CP 0DH           ;ENTER?
050B 282E     08060          JR Z,FIRST
050D FE08     08070          CP 08H           ;GO  BACK
050F 2826     08080          JR Z.BACKW       ;RETURN PRINT ADDRESS
0511 FE20     08090          CP 20H           ;WILL NOT ALTER DATA
0513 2813     08100          JR Z,INCK        ;INCREMENT AND KEEP
0515 CDB201   08110          CALL CRT
0518 FE2F     08120          CP '/'           ;NEW ADDRESS
051A 2818     08130          JR Z,NADDS       ;LAST STRING WAS ADDRESS
051C CD5D02   08140          CALL VALHEX      ;IS IT HEX
051F 38E0     08150          JR C,PRINIT      ;TRY AGAIN
0521 04       08160          INC B
0522 CD9701   08170          CALL LDHL        ;SHIFT INTO HL PAIR
0525 C30605   08180          JP INS3
0528 E1       08190 INCK     POP HL           ;GET OLD ADDRESS
0529 1803     08200          JR ENT1
052B 7D       08210 ENT      LD A,L           ;ONLY INFO IN L REQUIRED
052C E1       08220          POP HL
052D 77       08230          LD (HL),A
052E CD5C01   08240 ENT1     CALL SPCHL       ;SPACE FOLLOWED BY DATA
0531 23       08250          INC HL
0532 18C1     08260          JR INS1
0534 D1       08270 NADDS    POP DE
0535 18BE     08280          JR INS1
0537 E1       08290 BACKW    POP HL
0538 2B       08300          DEC HL
0539 18BA     08310          JR INS1          ;PRINT LAST ADDRESS
053B 10EE     08320 FIRST    DJNZ ENT
053D E1       08330          POP HL
053E C9       08340          RET
              08350          ;
              08360          ;VERIFY MEMORY
              08370          ;
053F CD0B02   08380 VERIFY   CALL G3N         ;GET ARGUMENTS
0542 1A       08390 VERIF1   LD A,(DE)
0543 EDA1     08400          CPI
0545 2813     08410          JR Z,VERF2       ;AGREES SO CONTINUE
0547 2B       08420          DEC HL           ;GO BACK FOR DISPLAY
0548 CD8001   08430          CALL ADDR        ;ADDRESS
054B CD5C01   08440          CALL SPCHL       ;AND DATA AT ADDRESS
054E EB       08450          EX DE,HL         ;NOW OTHER
054F CDA201   08460          CALL SPACE       ;SPACE
0552 CD8301   08470          CALL ADDR1       ;ADDRESS BUT NOT NEW LINE
0555 CD5C01   08480          CALL SPCHL       ;DATA AT OTHER ADDRESS
0558 EB       08490          EX DE,HL         ;RESTORE
0559 23       08500          INC HL           ;RESTORE HL
055A 13       08510 VERF2    INC DE           ;INC OTHER POINTER
055B E0       08520          RET PO           ;FINISHED?
055C 18E4     08530          JR VERIF1        ;CONTINUE
              08540          ;
              08550          ;EXAMINE REGISTERS
```

```
                08560           ;
055E CDEF02     08570 EXAM      CALL GET        ;NEXT IN STRING
0561 28FB       08580           JR Z,EXAM
0563 3821       08590           JR C,XALL       ;IF , EXAMINE ALL
0565 F5         08600           PUSH AF         ;SAVE REGISTER
0566 CDEF02     08610           CALL GET        ;PRIME OR NOT
0569 D627       08620           SUB '''         ;IF PRIME STORE 01H IN PRIMF
056B 2804       08630           JR Z,EXAM2      ;NO NEED TO CORRECT POINTER
056D FD2B       08640           DEC IY          ;ADJUST BUFFER POINTER
056F 3EFF       08650           LD A,0FFH       ;WHEN INC WILL BE 0
0571 3C         08660 EXAM2     INC A           ;IT IS NOW 01H
0572 324B40     08670           LD (PRIMF),A    ;FLAG REFLECTS PRIME REGISTER
0575 F1         08680 EXAM3     POP AF          ;GET REGISTER NAME
0576 214406     08690           LD HL,XTAB
0579 012000     08700           LD BC,32D       ;32 CHARACTERS AND DATA LOCATIONS
057C EDA1       08710 CONTCP    CPI             ;COMPARE (HL) TO A AND INC
057E CA1106     08720           JP Z,FOUND
0581 E20C03     08730           JP PO,ERROR     ;NOT IN TABLE
0584 18F6       08740           JR CONTCP       ;CONTINUE COMPARISON
0586 AF         08750 XALL      XOR A           ;CLEAR A
0587 324B40     08760           LD (PRIMF),A    ;MUST BE ZERO
058A 216206     08770           LD HL,PCP       ;PROGRAM COUNTER IN XTAB
058D CDC605     08780           CALL PVAL       ;PRINT 'P' AND CONTENTS OF PC
0590 CDA201     08790           CALL SPACE
0593 214406     08800           LD HL,XTAB      ;REGISTER TABLE
0596 0606       08810           LD B,6H         ;REST OF REGISTERS
0598 CDB905     08820           CALL ALL1       ;DISPLAY S,IX,IY,N,I,V
059B 0609       08830 ALL3      LD B,9H         ;NINE REGISTER DATA DISPLAY
059D CD9E01     08840           CALL NEWLIN
05A0 CDB905     08850           CALL ALL1
05A3 E5         08860           PUSH HL
05A4 C5         08870           PUSH BC
05A5 CDFA05     08880           CALL FLAGP      ;PRINT FLAG DATA
05A8 C1         08890           POP BC
05A9 E1         08900           POP HL
05AA 3A4B40     08910           LD A,(PRIMF)
05AD 3C         08920           INC A
05AE 324B40     08930           LD (PRIMF),A
05B1 FE02       08940           CP 02H
05B3 C8         08950           RET Z
05B4 215006     08960           LD HL,RA
05B7 18E2       08970           JR ALL3
05B9 C5         08980 ALL1      PUSH BC
05BA CDC605     08990           CALL PVAL       ;PRINT NAME AND VALUE
05BD F5         09000           PUSH AF
05BE CDA201     09010           CALL SPACE
05C1 F1         09020           POP AF
05C2 C1         09030           POP BC
05C3 10F4       09040           DJNZ ALL1       ;REST OF REGISTERS
05C5 C9         09050           RET
05C6 46         09060 PVAL      LD B,(HL)       ;HL POINTS TO REG NAME
05C7 23         09070           INC HL          ;POINT TO BIAS
05C8 7E         09080           LD A,(HL)       ;SIGN WILL HAVE DATA AS TO 1 OR 2
05C9 112440     09090 PVAL1     LD DE,SSTACK
05CC 83         09100           ADD A,E
05CD 5F         09110           LD E,A
05CE 7A         09120           LD A,D          ;GET HIGH ORDER
```

```
05CF CE00     09130          ADC A,00H       ;PAGE BOUNDRY
05D1 57       09140          LD D,A          ;DOUBLE BYTE ADD
05D2 23       09150          INC HL          ;POINT TO NEXT ENTRY
05D3 78       09160          LD A,B          ;PRINT NAME
05D4 CDB201   09170          CALL CRT
05D7 3A4B40   09180          LD A,(PRIMF)
05DA B7       09190          OR A
05DB 2808     09200          JR Z,NPRIM
05DD 7B       09210          LD A,E
05DE D608     09220          SUB 8H
05E0 5F       09230          LD E,A
05E1 3E27     09240          LD A,'''
05E3 1802     09250          JR PRIME
05E5 3E3D     09260 NPRIM    LD A,'='
05E7 CDB201   09270 PRIME    CALL CRT
05EA CB7B     09280          BIT 7,E         ;CHECK FOR ONE OR TWO BYTE NUMBER
05EC 2807     09290          JR Z,ONEB       ;NO ADJUST IF ONE BYTE
05EE CBBB     09300          RES 7,E         ;TWO BYTE, CORRECT BIAS
05F0 1A       09310          LD A,(DE)       ;GET VALUE
05F1 CD6001   09320          CALL PHEX       ;PRINT IT
05F4 1B       09330          DEC DE          ;DECREMENT POINTER
05F5 1A       09340 ONEB     LD A,(DE)       ;GET VALUE
05F6 CD6001   09350          CALL PHEX
05F9 C9       09360          RET
05FA 47       09370 FLAGP    LD B,A
05FB CDA201   09380          CALL SPACE
05FE 213C06   09390          LD HL,FTAB
0601 CB20     09400 FLAG1    SLA B
0603 3805     09410          JR C,OFLAG
0605 C8       09420          RET Z
0606 3E20     09430          LD A,20H
0608 1801     09440          JR FLAG2
060A 7E       09450 OFLAG    LD A,(HL)
060B CDB201   09460 FLAG2    CALL CRT
060E 23       09470          INC HL
060F 18F0     09480          JR FLAG1
0611 4E       09490 FOUND    LD C,(HL)       ;SAVE BIAS AND NUMBER OF BYTES
0612 47       09500          LD B,A          ;B REGISTER GETS NAME
0613 79       09510          LD A,C          ;BIAS INTO AC
0614 CDC905   09520          CALL PVAL1      ;WRITE FORMER CONTENTS
0617 CDEF02   09530 FOUND1   CALL GET        ;DATA TO REPLACE OR END
061A 3811     09540          JR C,NOTTWO     ;NEXT IN STRING
061C 28F9     09550          JR Z,FOUND1     ;SKIP BLANK (20H)
061E FD2B     09560          DEC IY          ;ADJUST BUFFER POINTER GHL WANTS
0620 CD4302   09570          CALL GHL        ;VALID HEX IN FIRST POSITION
0623 7D       09580          LD A,L          ;MOVE TO STACK
0624 12       09590          LD (DE),A       ;IF ONLY ONE ITS IN L
0625 79       09600          LD A,C          ;SIGN HAS 1,2 INFORMATION
0626 B7       09610          OR A            ;SET FLAG
0627 F22D06   09620          JP P,NOTTWO     ;IF POSITIVE ONLY ONE BYTE
062A 13       09630          INC DE
062B 7C       09640          LD A,H          ;GET HIGH DATA
062C 12       09650          LD (DE),A
062D CDA201   09660 NOTTWO   CALL SPACE
0630 79       09670          LD A,C          ;GET REGISTER INFORMATION
0631 CDC905   09680          CALL PVAL1
0634 4F       09690          LD C,A          ;DATA LAST PRINTED INTO C
```

```
0635 78       09700          LD A,B          ;CHECK FOR FLAG
0636 FE46     09710          CP 'F'          ;IF FLAG PRINT FLAGS
0638 79       09720          LD A,C          ;DATA BACK TO A FOR FLAG PRINT
0639 28BF     09730          JR Z,FLAGP
063B C9       09740          RET             ;RETURN TO STRING PROCESS
063C 53       09750 FTAB     DEFM 'SZXHX/NC'
0644 53       09760 XTAB     DEFM 'S'
0645 81       09770          DEFB 81H
0646 58       09780          DEFM 'X'
0647 83       09790          DEFB 83H
0648 59       09800          DEFM 'Y'
0649 85       09810          DEFB 85H
064A 49       09820          DEFM 'I'
064B 07       09830          DEFB 07H
064C 4E       09840          DEFM 'N'
064D 06       09850          DEFB 06H
064E 56       09860          DEFM 'V'
064F 9B       09870          DEFB 9BH
0650 41       09880 RA       DEFM 'A'
0651 11       09890          DEFB 11H
0652 42       09900          DEFM 'B'
0653 13       09910          DEFB 13H
0654 43       09920          DEFM 'C'
0655 12       09930          DEFB 12H
0656 44       09940          DEFM 'D'
0657 15       09950          DEFB 15H
0658 45       09960          DEFM 'E'
0659 14       09970          DEFB 14H
065A 48       09980          DEFM 'H'
065B 17       09990          DEFB 17H
065C 4C       10000          DEFM 'L'
065D 16       10010          DEFB 16H
065E 4D       10020          DEFM 'M'
065F 97       10030          DEFB 97H
0660 46       10040          DEFM 'F'
0661 10       10050          DEFB 10H
0662 50       10060 PCP      DEFM 'P'
0663 99       10070          DEFB 99H
              10080 ;
              10090 ;        CASSETTE WRITE ROUTINE
              10100 ;
0664 CD0B02   10110 WRIT     CALL G3N
0667 D5       10120          PUSH DE
0668 E5       10130          PUSH HL
0669 C5       10140          PUSH BC
066A 3E04     10150 GNAM3    LD A,4H
066C D3FF     10160          OUT (0FFH),A
066E 0600     10170          LD B,00H
0670 10FE     10180 DT       DJNZ DT
0672 78       10190          LD A,B          ;ZERO ACCUMULATOR
0673 CDEC06   10200 LEAD     CALL TOUT
0676 10FB     10210          DJNZ LEAD
0678 3EA5     10220 WRIT1    LD A,0A5H       ;SYNC MARK
067A CDEC06   10230          CALL TOUT
067D 3E55     10240          LD A,55H        ;FILE NAME MARK
067F CDEC06   10250          CALL TOUT       ;WRITE NAME HEADER
0682 0606     10260          LD B,06H        ;6 BYTE NAME
```

```
0684 CDF102    10270          CALL GET1        ;LAST ENTRY
0687 380E      10280          JR C,FINB        ;IF COMMA NO NAME FILL BLANKS
0689 CDEF02    10290 WNAME    CALL GET         ;NEXT ENTRY
068C 3809      10300          JR C,FINB        ;FINISH WITH BLANKS
068E CDEC06    10310          CALL TOUT        ;WRITE NAME
0691 10F6      10320          DJNZ WNAME
0693 FD23      10330          INC IY           ;ADJUST BUFFER
0695 1807      10340          JR WRIT3         ;FULL NAME
0697 3E20      10350 FINB     LD A,20H         ;FINISH WITH BLANKS
0699 CDEC06    10360          CALL TOUT
069C 10F9      10370          DJNZ FINB
069E D1        10380 WRIT3    POP DE
069F E1        10390          POP HL
06A0 15        10400 WRIT2    DEC D
06A1 FAD006    10410          JP M,FINISH
06A4 AF        10420          XOR A
06A5 CDAA06    10430          CALL TDATA
06A8 18F6      10440          JR WRIT2
06AA 47        10450 TDATA    LD B,A
06AB 3E3C      10460          LD A,3CH
06AD CDEC06    10470          CALL TOUT
06B0 78        10480          LD A,B
06B1 CDEC06    10490          CALL TOUT
06B4 7D        10500          LD A,L
06B5 CDEC06    10510          CALL TOUT
06B8 7C        10520          LD A,H
06B9 CDEC06    10530          CALL TOUT
06BC 7C        10540          LD A,H
06BD 22FE3F    10550          LD (3FFEH),HL    ;SHOW SOMETHING
06C0 7C        10560          LD A,H
06C1 85        10570          ADD A,L          ;START OF CHECKSUM
06C2 4F        10580          LD C,A           ;CHECKSUM STORAGE
06C3 7E        10590 TDATA1   LD A,(HL)
06C4 CDEC06    10600          CALL TOUT
06C7 7E        10610          LD A,(HL)
06C8 81        10620          ADD A,C
06C9 4F        10630          LD C,A
06CA 23        10640          INC HL
06CB 10F6      10650          DJNZ TDATA1
06CD 79        10660          LD A,C
06CE 181C      10670          JR TOUT
06D0 AF        10680 FINISH   XOR A
06D1 BB        10690          CP E
06D2 2804      10700          JR Z,WRIT5       ;NOTHING TO WRITE ,EXCEPT HEAD
06D4 7B        10710          LD A,E           ;MORE TO COME
06D5 CDAA06    10720          CALL TDATA
06D8 3E78      10730 WRIT5    LD A,78H         ;LAST BLOCK HEADER
06DA CDEC06    10740          CALL TOUT
06DD E1        10750          POP HL           ;EXECUTE ADDRESS
06DE 7D        10760          LD A,L
06DF CDEC06    10770          CALL TOUT
06E2 7C        10780          LD A,H
06E3 CDEC06    10790          CALL TOUT
06E6 CD7901    10800          CALL PNHL        ;SHOW ADDRESS
06E9 C32103    10810          JP TOFF1         ;TURN TAPE OFF
06EC D9        10820 TOUT     EXX              ;SAVE REGISTERS
06ED 0E08      10830          LD C,8
```

```
06EF 57       10840          LD D,A
06F0 37       10850 TOUT1    SCF              ;CARRY WILL OUPTUT A PULSE
06F1 CD0607   10860          CALL PULSE       ;OUT SYNC PULSE
06F4 7A       10870          LD A,D           ;GET DATA BIT
06F5 07       10880          RLCA             ;PUT IN CARRY
06F6 57       10890          LD D,A           ;RETURN TO D REGISTER
06F7 CD0607   10900          CALL PULSE       ;OUT WHATEVER
06FA 0D       10910          DEC C
06FB 20F3     10920          JR NZ,TOUT1
06FD 3A8038   10930          LD A,(3880H)     ;SHIFT PRESSED
0700 B7       10940          OR A             ;SET FLAG
0701 D9       10950          EXX              ;RETURN REGISTERS
0702 C22103   10960          JP NZ,TOFF1      ;IF NON-ZERO GET OUT
0705 C9       10970          RET
0706 3E02     10980 PULSE    LD A,02H         ;WHEN RLA 04 WHICH IS OUT 00
0708 17       10990          RLA              :OR 01 WITH CASSETTE ON
0709 D3FF     11000          OUT (0FFH),A
070B 0614     11010          LD B,20D         ;SHORT DELAY
070D 10FE     11020 P1       DJNZ P1
070F E603     11030          AND 03H          ;KEEP INFORMATION BUT NO
0711 17       11040          RLA              ;CARRY
0712 F604     11050          OR 04H           ;KEEP RECORDER ON
0714 D3FF     11060          OUT (0FFH),A     ;IF 1 ANOTHER PULSE
0716 0614     11070          LD B,20D         ;DELAY
0718 10FE     11080 P2       DJNZ P2
071A 3E04     11090          LD A,04H
071C D3FF     11100          OUT (0FFH),A     ;RETURN TO ZERO LEVEL
071E 065E     11110          LD B,94D         ;REMAINDER OF MILLISECOND
0720 10FE     11120 P3       DJNZ P3
0722 C9       11130          RET
              11140 ;
              11150 ;        CASSETTE READ ROUTINE
              11160 ;
0723 CDEF02   11170 READ     CALL GET
0726 28FB     11180          JR Z,READ        ;SKIP SPACES
0728 3807     11190          JR C,RD11
072A FD2B     11200          DEC IY           ;ADJUST
072C CD4302   11210          CALL GHL
072F 1803     11220          JR RD22
0731 210000   11230 RD11     LD HL,00H
0734 E5       11240 RD22     PUSH HL          ;WILL POP TO DE
0735 3E04     11250          LD A,4H
0737 D3FF     11260          OUT (0FFH),A     ;TURN ON TAPE
0739 FD2B     11270          DEC IY           ;ADJUST
073B CDAE07   11280 RD1      CALL BIT
073E FEA5     11290          CP 0A5H          ;SYNC?
0740 20F9     11300          JR NZ,RD1
0742 2A4040   11310          LD HL,(CURSOR)   ;GET CURSOR WANT LOAD FAST
0745 0606     11320          LD B,6H
0747 CDA207   11330          CALL BYTE
074A FE55     11340          CP 55H
074C 2039     11350          JR NZ,TOFF
074E CDA207   11360 RD2      CALL BYTE
0751 77       11370          LD (HL),A
0752 23       11380          INC HL           ;ADJUST POINTER AND PROVIDE SPACE
0753 10F9     11390          DJNZ RD2
0755 23       11400          INC HL           ;SPACE
```

```
0756 224040    11410          LD (CURSOR),HL
0759 CDA207    11420 RD3      CALL BYTE
075C FE78      11430          CP 78H
075E 282A      11440          JR Z,END1
0760 FE3C      11450          CP 03CH
0762 2023      11460          JR NZ,TOFF
0764 CDA207    11470          CALL BYTE
0767 47        11480          LD B,A
0768 CDA207    11490          CALL BYTE
076B 6F        11500          LD L,A
076C CDA207    11510          CALL BYTE
076F 67        11520          LD H,A
0770 22FE3F    11530          LD (3FFEH),HL
0773 85        11540          ADD A,L          ;START OF CHECKSUM
0774 4F        11550          LD C,A
0775 D1        11560          POP DE           ;GET BIAS FOR LOAD
0776 D5        11570          PUSH DE
0777 19        11580          ADD HL,DE        ;BIAS ADDED TO HL
0778 CDA207    11590 RD4      CALL BYTE
077B 77        11600          LD (HL),A
077C 23        11610          INC HL
077D 81        11620          ADD A,C
077E 4F        11630          LD C,A
077F 10F7      11640          DJNZ RD4
0781 CDA207    11650          CALL BYTE        ;GET CHECKSUM
0784 B9        11660          CP C
0785 28D2      11670          JR Z,RD3
0787 C30C03    11680 TOFF     JP ERROR
078A CDA207    11690 END1     CALL BYTE
078D 6F        11700          LD L,A
078E CDA207    11710          CALL BYTE
0791 67        11720          LD H,A           ;HL HAS START ADDRESS
0792 D1        11730          POP DE
0793 110000    11740          LD DE,00H        ;CORRECT STACK AND ZERO
0796 19        11750          ADD HL,DE
0797 CD7901    11760          CALL PNHL
079A 223C40    11770          LD (PCSAVE),HL   ;STORE TO PC PLACE
079D FD23      11780          INC IY           ;ADJUST BUFFER
079F C32103    11790          JP TOFF1         ;TURNTAPE OFF
07A2 C5        11800 BYTE     PUSH BC
07A3 E5        11810          PUSH HL
07A4 0608      11820          LD B,8H
07A6 CDAE07    11830 BYTE1    CALL BIT
07A9 10FB      11840          DJNZ BYTE1
07AB E1        11850          POP HL
07AC C1        11860          POP DC
07AD C9        11870          RET
07AE C5        11880 BIT      PUSH BC
07AF F5        11890          PUSH AF
07B0 3E04      11900          LD A,4H
07B2 D3FF      11910          OUT (0FFH),A     ;RESET LATCH BEFORE READ
07B4 3A8038    11920 BIT1     LD A,(3880H)
07B7 B7        11930          OR A             ;SEE IF ESCAPE
07B8 20CD      11940          JR NZ,TOFF
07BA DBFF      11950 BIT5     IN A,(0FFH)
07BC 17        11960          RLA
07BD 30F5      11970          JR NC,BIT1       ;WAIT FOR PULSE
```

```
07BF 0640      11980          LD B,40H         ;SHORT DELAY
07C1 10FE      11990 BIT2     DJNZ BIT2
07C3 3E04      12000          LD A,4H
07C5 D3FF      12010          OUT (0FFH),A     ;RESET LATCH
07C7 0680      12020          LD B,80H         ;WINDOW TO LOOK FOR PULSE
07C9 10FE      12030 BIT3     DJNZ BIT3
07CB DBFF      12040          IN A,(0FFH)      ;IN DATA A 1 IS IN POSITION (7)
07CD 4F        12050          LD C,A
07CE F1        12060          POP AF
07CF CB11      12070          RL C
07D1 17        12080          RLA              ;ONE BIT ROTATE THROUGH AC
07D2 C1        12090          POP BC
07D3 C9        12100          RET
               12110          ;
               12120          ;FIND ONE OR TWO BYTES
               12130          ;
07D4 3E02      12140 FINDTO   LD A,02H         ;FLAG FOR FINDING TWO BYTES
07D6 1802      12150          JR FIND
07D8 3E01      12160 FINDON   LD A,01H         ;FLAG FOR FINDING ONE BYTE
07DA 324340    12170 FIND     LD (COUNT),A     ;SAVE FOR REFERENCE
07DD CD0B02    12180          CALL G3N         ;GET ARGUMENTS
07E0 7B        12190 FND1     LD A,E           ;LOW ORDER BYTE OR ONLY BYTE
07E1 EDA1      12200 FND2     CPI              ;IS IT THERE
07E3 E0        12210          RET PO           ;FINISHED  AND NOT FOUND
07E4 20FB      12220          JR NZ,FND2       ;NOT THERE BUT CONTINUE SEARCH
07E6 3A4340    12230          LD A,(COUNT)     ;MORE THAN ONE
07E9 3D        12240          DEC A            ;IF ONE NOW ITS ZERO
07EA 2806      12250          JR Z,SHOW
07EC 7A        12260          LD A,D           ;NEED SECOND BYTE
07ED BE        12270          CP (HL)          ;IT MUST BE NEXT. THE CPI INC HL
07EE 2802      12280          JR Z,SHOW        ;IT WAS THERE SO SHOW IT
07F0 18EE      12290          JR FND1          ;CONTINUE SEARCH START WITH FIRST
07F2 3A4340    12300 SHOW     LD A,(COUNT)     ;GET NUMBER, DESTROYED BY DEC
07F5 C5        12310          PUSH BC          ;SAVE
07F6 4F        12320          LD C,A           ;PREPARE FOR DISPLAY
07F7 0600      12330          LD B,00H
07F9 2B        12340          DEC HL           ;ADJUST HL
07FA CDA703    12350          CALL DISP1       ;SHOW ADDRESS FOLLOWED BY DATA
07FD C1        12360          POP BC           ;GET BYTES REMAINING
07FE 18E0      12370          JR FND1          ;CONTINUE GET ALL OCCURRENCES
               12380          ;
0000           12390          END
00000 TOTAL ERRORS


ADDR   0180 02530   05590 05930 07760 07970 08430
ADDR1  0183 02540   07180 08470
ADDR2  0186 02550   07330
ADJUST 0175 02440   02390
ALL1   05B9 08980   08820 08850 09040
ALL3   059B 08830   08970
ASCII  0363 05570   05250
ASCII1 0366 05580   05660
ASCII2 036E 05610   05670
ASCII4 036F 05620
BACKW  0537 08290   08080
BIT    07AE 11880   11280 11830
```

```
BIT1    07B4 11920   11970
BIT2    07C1 11990   11990
BIT3    07C9 12030   12030
BIT5    07BA 11950
BKSP    01E7 03210   03070
BOTMS   4026 00220   06800
BPNO    4045 00160   01730 01920 06600
BPOINT  4048 00170   01630 06730
BREAK   0403 06460   06410
BREAK1  0413 06530   06660 06720
BREAKS  4080 00250   01810 06350
BUF     0127 01970   00930
BUF0    012C 01990
BUF1    012F 02010   02120 02170 02240
BUFFER  4080 00090   00250 00910 05120 05760 05840
BUFFR   037A 05710   05260
BYTE    07A2 11800   11330 11360 11420 11470 11490 11510 11590
                     11650 11690 11710
BYTE1   07A6 11830   11840
CAPIT   02C8 04630   04610
CD      008A 01050   01060 05880
CHAR    02D7 04720   04700
CHKIN   027E 04140   02840 04050
CLEAR   0203 03370   03190 03410
CLEARS  01FE 03350   00790 05270
CLRBP   0107 01720   00890 01660 05140
CLRBP1  011D 01870   01900
CMARK   4044 00150   00840 03250
CMND    0080 01010   05060
COM1    006F 00910   06240
COMAND  006C 00900   04940
COMTAB  032D 05240   01090
CON     002B 00580   00550
CON1    0033 00620   00590
CON2    004C 00760   00630
CONTCP  057C 08710   08740
CONTI1  0302 05030   05050
CONTIN  0304 05040   01020 02860 05200
COUNT   4043 00140   12170 12230 12300
CRT     01B2 02930   02160 02220 02640 05620 08030 08110 09170
                     09270 09460
CRT1    01BD 03000   02980
CURSCR  4040 00120   03020 03240 11310 11410
D1      028A 04210   04210
DELETE  013E 02080
DISP1   03A7 05930   01710 05990 12350
DISP2   03AA 05940   06000
DISPLY  03A4 05920   05280
DLT     0142 02100
DLTE    0147 02130   02200
DLTE1   0148 02140
DT      0670 10180   10180
END1    078A 11690   11440
ENT     052B 08210   08320
ENT1    052E 08240   08200
ENTER   03C6 06180   05310 05440
ER0     0311 05090   05110
```

```
ER1     0318 05120    05100
ER2     0329 05190    05170
ERROR   030C 05070    01080 01120 03600 03800 05340 05350 05490
                      08730 11680
EXAM    055E 08570    05480 08580
EXAM2   0571 08660    08630
EXAM3   0575 08680
EXX     00CF 01450    01390
FILL    03B7 06060    05240
FILL1   03BB 06080    00780 06110
FINB    0697 10350    10280 10300 10370
FIND    07DA 12170    12150
FINDON  07D8 12160    05300
FINDTO  07D4 12140    05410
FINISH  06D0 10680    10410
FIRST   053B 08320    08060
FLAG1   0601 09400    09480
FLAG2   060B 09460    09440
FLAGP   05FA 09370    08880 09730
FND1    07E0 12190    12290 12370
FND2    07E1 12200    12220
FOUND   0611 09490    08720
FOUND1  0617 09530    09550
FTAB    063C 09750    09390
FULSCN  01D1 03120    03100
G2N     0214 03500    03450 05570 05920 07280
G2N0    021D 03540    03550
G2N1    0233 03630    03570
G2N2    023A 03670    03620 03720
G2N3    023E 03710    03530
G3N     020B 03450    06060 07550 08380 10110 12180
GBYTE   026E 04040    02010 04080 04110 08040
GET     02EF 04910    01050 03540 03770 03820 05090 05710 06370
                      06460 08570 08610 09530 10290 11170
GET1    02F1 04920    03520 03590 05040 06220 06360 07790 07870
                      10270
GET3    02FF 04990
GHL     0243 03760    03470 03500 03610 03630 06430 06520 06620
                      06670 07130 07150 07670 07690 07750 07780
                      07960 09570 11210
GHL0    0246 03770    03780
GHL1    0251 03810    03840
GNAM3   066A 10150
GO      03D9 06290    05290
GO1     03EB 06360    06450 06610
GO2     03EE 06370    06380
GO4     0459 06870    06850
HDATA   02A6 04400    04340 04430
HEXM    0478 07130    05320
HLSAVE  403A 00240    01270 01330 06980
IN      048D 07280    05380
INCK    0528 08190    08100
INNEXT  0492 07310    07420
INS1    04F5 07970    08260 08280 08310
INS2    04F8 07980
INS3    0506 08040    08180
INSERT  04F2 07960    05430
```

```
JMP     42A0 00350    06750 06770 06790 07070 07090
KYD1    0294 04300    04180 04220
LDHL    0197 02700    03810 08170
LDREG   0436 06730    06390
LEAD    0673 10200    10210
LINEF   01D5 03140    03050
LIST    04AC 07470    05360
LOWBIT  02AC 04440    04410
MINUS   042A 06670    06500
MONSP   42C0 00340    00350 00880 01010 01670
MOV1    04BA 07560
MOVE    04B7 07550    05370
NADDS   0534 08270    08130
NALPH1  02CF 04670    04760 04800
NALPHA  02CB 04650    04560
NEWAD1  04ED 07900    07810
NEWADR  04EF 07910
NEWLIN  019E 02800    01040 02530 08840
NEXT    029A 04320    04370
NOT7    02DA 04740    04660
NOTTWO  062D 09660    09540 09620
NPRIM   05E5 09260    09200
OFLAG   060A 09450    09410
OLD     4047 00270    04060
ONEB    05F5 09340    09290
OUT     04C5 07670    05390
OUT2    01EB 03240    03130 03220
P1      070D 11020    11020
P2      0718 11080    11080
P3      0720 11120    11120
PCH     0150 02180    02070
PCHAR   0152 02190    02090
PCP     0662 10060    08770
PCSAVE  403C 00230    00240 00290 01250 01420 01440 01690 06440
                      06640 06690 06780 11770
PHEX    0160 02310    02500 02520 07320 07360 09320 09350
PHEX1   0170 02410    02450
PHEXH   0163 02320    02310
PHEXL   0167 02360
PLUS    0420 06620    06480
PMSG    018D 02610    02660 05130
PMSG1   018A 02600    00900
PNHL    0179 02490    02540 07230 10800 11760
POPAF   0474 07080    07020
PPHL    015F 02300
PRIME   05E7 09270    09250
PRIMF   404B 00190    08670 08760 08910 08930 09180
PRINIT  0501 08020    08150
PRNT    01A4 02830    02410 02560 02810
PRNT1   01A5 02840    02880
PRNTFG  404A 00180    03290 07480 07500
PROM    401B 00310    05400
PULSE   0706 10980    10860 10900
PVAL    05C6 09060    08780 08990
PVAL1   05C9 09090    09520 09680
RA      0650 09880    08960
RD1     073B 11280    11300
```

```
RD11    0731 11230    11190
RD2     074E 11360    11390
RD22    0734 11240    11220
RD3     0759 11420    11670
RD4     0778 11590    11640
READ    0723 11170    05420 11180
REF     4000 00080    00090 00100 00110 00200 00270 00280 00300
                      00310 00320 00330 00410
REF1    4040 00110    00120 00130 00140 00150 00160 00170 00180
                      00190
REF2    4024 00200    00210 00230
RSAVE   00A6 01230    06330
RST     00FF 00070    06560
RSTART  0066 00880    02030
RSTLOC  4012 00060    06310 06340
S1      000B 00440    00420
S2      0013 00470    00450 00560
S3      001B 00510    00490
S4      0023 00550    00530
SAVE    0385 05760    05740
SAVE1   00DE 01560    01540
SCREEN  403E 00290    00810 01610 06180
SHOW    07F2 12300    12250 12280
SIGNON  0043 00730    02600
SPACE   01A2 02820    02290 05600 07340 07370 07380 08460 08790
                      09010 09380 09660
SPCHL   015C 02290    05940 07850 07980 08240 08440 08480
SPEC    02E7 04810    04450
SSFLAG  4042 00130    01370 01410 06160 07000
SSTACK  4024 00210    00220 00820 01310 06960 09090
STEP    03C1 06150    05500
SUBS    04D2 07760    07920
SUBS1   04D5 07770    07890
SUBS3   04E0 07820
SUBSM   04CF 07750    05330
TABLE   003B 00650    04810
TDATA   06AA 10450    10430 10720
TDATA1  06C3 10590    10650
TEM     4180 00100    05780 05820
TOFF    0787 11680    11350 11460 11940
TOFF1   0321 05150    10810 10960 11790
TOUT    06EC 10820    10200 10230 10250 10310 10360 10470 10490
                      10510 10530 10600 10670 10740 10770 10790
TOUT1   06F0 10850    10920
TRAN    0393 05820    05720
TTYP    4018 00300    00770 03310
TWO     0284 04180    04240
UCFLAG  4046 00280    01980 04570
USER    401E 00320    05450
USERSP  4300 00330    00340 00620
VALHEX  025D 03890    03790 03830 08140
VERF2   055A 08510    08410
VERIF1  0542 08390    07630 08530
VERIFY  053F 08380    05460
VIDEO   3C00 00260    03150 03160 03200 03350
WNAME   0689 10290    10320
WRIT    0664 10110    05470
```

```
WRITT1   0678 10220
WRITT2   06A0 10400   10440
WRITT3   069E 10380   10340
WRITT5   06D8 10730   10700
XALL     0586 08750   01680 08590
XTAB     0644 09760   08690 08800
```

APPENDIX E

Source Listing for FROLIC Monitor for the Model III TRS-80

```
                00010 ;
                00020 ;         MODEL III TRS-80 DEVELOPMENT SYSTEM
                00030 ;         DISK NAME MODIII/ASM
                00040 ;         MODEL III FROLIC MONITOR
                00050 ;
4012            00060 RSTLOC    EQU 4012H
00FF            00070 RST       EQU 0FFH
4000            00080 REF       EQU 04000H      ;START OF RAM
4080            00090 BUFFER    EQU REF+80H
4180            00100 TEM       EQU REF+180H    ;SAVED BUFFER AREA
4040            00110 REF1      EQU REF+40H
4040            00120 CURSOR    EQU REF1
4042            00130 SSFLAG    EQU REF1+2      ;SINGLE STEP FLAG
4043            00140 COUNT     EQU REF1+3H
4044            00150 CMARK     EQU REF1+4H
4045            00160 BPNO      EQU REF1+5H
4048            00170 BPOINT    EQU REF1+8H     ;BUFFER POINTER
404A            00180 PRNTFG    EQU REF1+0AH
404B            00190 PRIMF     EQU REF1+0BH
4024            00200 REF2      EQU REF+24H
4024            00210 SSTACK    EQU REF2
4026            00220 BOTMS     EQU SSTACK+2H
403C            00230 PCSAVE    EQU REF2+18H
403A            00240 HLSAVE    EQU PCSAVE-2
4080            00250 BREAKS    EQU BUFFER      ;JUST BELOW BUFFER
3C00            00260 VIDEO     EQU 3C00H
4047            00270 OLD       EQU REF+47H
4046            00280 UCFLAG    EQU REF+46H
403E            00290 SCREEN    EQU PCSAVE+2H
4018            00300 TTYP      EQU REF+18H
401B            00310 PROM      EQU REF+1BH
401E            00320 USER      EQU REF+1EH
4300            00330 USERSP    EQU REF+300H    ;USER STACK POINTER
42C0            00340 MONSP     EQU USERSP-40H  ;MONITOR STACKPOINTER
42A0            00350 JMP       EQU MONSP-20H   ;TOP OF MONITOR STACK
                00360           ;
0000            00370           ORG 0000H       ;*** FOR SYSTEM LOAD ***
                00380           ;
0000 AF         00390           XOR A           ;CLEAR AC
0001 D3EC       00400           OUT (0ECH),A    ;LOWER CASE AND CASSETTE OFF
0003 210040     00410           LD HL,REF       ;SYSTEM RAM
0006 1803       00420           JR S1           ;BY PASS RESTARTS
0008 C30040     00430           JP 4000H        ;RST 1
000B 010003     00440 S1        LD BC,0300H     ;CLEAR 1ST 3 PAGES
000E 1803       00450           JR S2
0010 C30340     00460           JP 4003H        ;RST 2
0013 77         00470 S2        LD (HL),A
0014 ED56       00480           IM 1            ;SET INTERRUPT MODE 1
0016 1803       00490           JR S3
0018 C30640     00500           JP 4006H        ;RST 3
001B EDA1       00510 S3        CPI
001D F3         00520           DI              ;DISABLE INTERRUPTS
001E 1803       00530           JR S4
0020 C30940     00540           JP 4009H        ;RST 4
0023 E22B00     00550 S4        JP PO,CON       ;CONTINUE INITIALIZATION
0026 18EB       00560           JR S2           ;CONTINUE TO CLEAR RAM
0028 C30C40     00570           JP 400CH        ;RST 5
```

```
002B 3EC9       00580 CON     LD A,0C9H
002D 1804       00590         JR CON1
002F 00         00600         NOP               ;FILL
0030 C30F40     00610         JP 400FH          ;RST 6
0033 310043     00620 CON1    LD SP,USERSP      ;SET USER STACK
0036 1814       00630         JR CON2
0038 C31240     00640         JP 4012H          ;RST 7
003B 0D         00650 TABLE   DEFB 0DH
003C 0C         00660         DEFB 0CH
003D 01         00670         DEFB 01H
003E 0B         00680         DEFB 0BH
003F 0A         00690         DEFB 0AH
0040 08         00700         DEFB 08H
0041 09         00710         DEFB 09H
0042 20         00720         DEFB 20H
0043 0D         00730 SIGNON  DEFB 0DH          ;WILL INITIALIZE CURSOR
0044 46         00740         DEFM 'FROLIC:'
004B 01         00750         DEFB 01H          ;END OF MESSAGE
004C 0E09       00760 CON2    LD C,09           ;LOAD RETURNS IN USER RAM
004E 211840     00770         LD HL,TTYP        ;START OF FILL
0051 CDA903     00780         CALL FILL1
0054 CDED01     00790         CALL CLEARS       ;CLEAR SCREEN ONLY ON POWER UP
0057 21003C     00800         LD HL,3C00H
005A 223E40     00810         LD (SCREEN),HL
005D ED732440   00820         LD (SSTACK),SP    ;INITIALIZE USER STACK
0061 3EA0       00830         LD A,0A0H         ;CURSOR MARK
0063 324440     00840         LD (CMARK),A      ;PLACE IN RAM
                00850 ;
                00860 ;       RESET START OF PROGRAM
                00870 ;
0066 31BE42     00880 RSTART  LD SP,MONSP-2     ;INITIALIZE STACK
0069 CD0701     00890         CALL CLRBP        ;CLEAR BREAKPOINTS
006C CD8A01     00900 COMAND  CALL PMSG1        ;FROLIC:
006F 218040     00910 COM1    LD HL,BUFFER      ;BUFFER BEGINNING
0072 E5         00920         PUSH HL           ;SAVE FOR LATER
0073 CD2701     00930         CALL BUF          ;BUFFER AVAILABLE FOR OTHERS
0076 362C       00940         LD (HL),2CH       ;DELIMITER
0078 23         00950         INC HL
0079 3601       00960         LD (HL),01H       ;BREAK AT END OF BUFFER
007B E1         00970         POP HL            ;GET BUFFER BEGINNING
007C 77         00980         LD (HL),A         ;PUT 0DH,ERRORS WILL DO LINEFEED
007D E5         00990         PUSH HL
007E FDE1       01000         POP IY            ;BUFFER POINTER FOR MONITOR
0080 31C042     01010 CMND    LD SP,MONSP       ;NECESSARY FOR COMMANDS
0083 21F802     01020         LD HL,CONTIN      ;RETURN ADDRESS ON STACK
0086 E5         01030         PUSH HL           ;PUT RETURN ADDRESS ON STACK
0087 CD9E01     01040         CALL NEWLIN       ;SAVES DOING IT IN COMMAND
008A CDE302     01050 CD      CALL GET          ;GET WILL GET FIRST NON ZERO CHAF
008D 28FB       01060         JR Z,CD           ;IGNORE LEADING SPACES
008F D640       01070         SUB '@'           ;BORROW IF < @
0091 DA0003     01080         JP C,ERROR        ;A 00 IF `
0094 211B03     01090         LD HL,COMTAB      ;POINT TO COMMAND TABLE
                01100                           ;A BORROW IS ALWAYS GENERATED
0097 FE5A       01110         CP 'Z'            ;BORROW VALID @ THRU Z
0099 D20003     01120         JP NC,ERROR
009C 87         01130         ADD A,A           ;DOUBLE NUMBER
009D 5F         01140         LD E,A            ;FOR DOUBLE ADD
```

```
009E 1600      01150          LD D,0H          ;BE AT A PAGE BOUNDARY
00A0 19        01160          ADD HL,DE        ;DAD WITH CARRY SET FROM COMPARE
00A1 5E        01170          LD E,(HL)        ;GET LOW BYTE
00A2 23        01180          INC HL
00A3 56        01190          LD D,(HL)        ;GET HIGH BYTE
00A4 EB        01200          EX DE,HL         ;JUMP TO ADDRESS
00A5 E9        01210          JP (HL)          ;EXECUTE REST OF COMMAND
               01220          ;
00A6 E3        01230 RSAVE    EX (SP),HL       ;GET PROGRAM COUNTER
00A7 2B        01240          DEC HL           ;ADJUST
00A8 223C40    01250          LD (PCSAVE),HL   ;AND SAVE
00AB E1        01260          POP HL
00AC 223A40    01270          LD (HLSAVE),HL   ;SAVE HL AND ADJUST STACKPOINTER
00AF F5        01280          PUSH AF          :SAVE FLAGS IN FORMER HL
00B0 210200    01290          LD HL,0002H      ;ADJUST FOR RST OR INT
00B3 39        01300          ADD HL,SP        ;GET SYSTEM USER STACK
00B4 222440    01310          LD (SSTACK),HL   ;SYSTEM STACK STORAGE
00B7 F1        01320          POP AF           ;GET FLAGS BACK
00B8 313A40    01330          LD SP,HLSAVE     ;START OF REGISTER SAVE
00BB D5        01340          PUSH DE
00BC C5        01350          PUSH BC
00BD F5        01360          PUSH AF          ;FLAGS ON STACK
00BE 3A4240    01370          LD A,(SSFLAG)    ;SINGLE STEP?
00C1 B7        01380          OR A             ;TEST
00C2 280B      01390          JR Z,EXX         ;NO SINGLE STEP IF ZERO
00C4 AF        01400          XOR A            :NOW ZERO
00C5 324240    01410          LD (SSFLAG),A    ;CORRECT FLAG
00C8 2A3C40    01420          LD HL,(PCSAVE)   ;GET PC COUNTER
00CB 23        01430          INC HL           :ADJUST
00CC 223C40    01440          LD (PCSAVE),HL   ;REPLACE
00CF D9        01450 EXX      EXX
00D0 E5        01460          PUSH HL
00D1 D5        01470          PUSH DE
00D2 C5        01480          PUSH BC
00D3 08        01490          EX AF,AF'
00D4 F5        01500          PUSH AF
00D5 ED57      01510          LD A,I           ;GET INT VECTOR AND IIF FLAG
00D7 67        01520          LD H,A           ;STORE IN H
00D8 2E00      01530          LD L,00H         ;INITIALIZE L
00DA E2DE00    01540          JP PO,SAVE1      ;IF PARITY FLAG NOT SET IFF INT D
00DD 2C        01550          INC L            :IFF SET INDICATE WITH NON ZERO
00DE E5        01560 SAVE1    PUSH HL          ;SAVE INT VECTOR AND INT SET FLAG
00DF FDE5      01570          PUSH IY
00E1 DDE5      01580          PUSH IX
00E3 21003C    01590          LD HL,3C00H      ;START OF SCREEN TRANSFERRED
00E6 010004    01600          LD BC,0400H
00E9 ED5B3E40  01610          LD DE,(SCREEN)   ;AREA STARTING AT SCREEN
00ED EDB0      01620          LDIR
00EF FD2A4840  01630          LD IY,(BPOINT)   ;GET LAST POSITION IN BUFFER
00F3 3B        01640          DEC SP           ;MUST NOT DESTROY STACK
00F4 3B        01650          DEC SP
00F5 CD0701    01660          CALL CLRBP       ;CLEAR BREAK POINTS
00F8 31BE42    01670          LD SP,MONSP-2    :CORRECT STACK
00FB CD7B05    01680          CALL XALL        ;DISPLAY REGISTERS
00FE 2A3C40    01690          LD HL,(PCSAVE)   ;GET PROGRAM COUNTER
0101 010500    01700          LD BC,05H        ;NUMBER OF BYTES
0104 C39503    01710          JP DISP1         ;DISPLAY
```

```
0107 210000    01720 CLRBP   LD HL,00H       ;CLEAR
010A 3A4540    01730         LD A,(BPNO)     ;LD HL WITH NO OF BPOINTS
010D B7        01740         OR A            ;SET ZERO FLAG
010E 6F        01750         LD L,A          ;STORE JUST IN CASE
010F C8        01760         RET Z           ;IF NO BREAKPOINTS GO TO COMMAND
0110 C1        01770         POP BC          ;GET RETURN ADDRESS
0111 29        01780         ADD HL,HL       ;DOUBLE VALUE IN L
0112 29        01790         ADD HL,HL       ;AND AGAIN
0113 EB        01800         EX DE,HL
0114 218040    01810         LD HL,BREAKS
0117 ED52      01820         SBC HL,DE       ;BORROW CLEAR DECREMENT STACK
0119 F9        01830         LD SP,HL        ;NEW STACK POSITION TOP
011A C5        01840         PUSH BC
011B E1        01850         POP HL          ;READY FOR RETURN
011C 47        01860         LD B,A          ;SHOULD STILL HAVE NUMBER
011D F1        01870 CLRBP1  POP AF
011E D1        01880         POP DE
011F 12        01890         LD (DE),A
0120 10FB      01900         DJNZ CLRBP1
0122 AF        01910         XOR A
0123 324540    01920         LD (BPNO),A     ;PUT ZERO IN BPNO
0126 E9        01930         JP (HL)         ;HL HAS RETURN ADDRESS
               01940         ;
               01950         ;FILL BUFFER WITH COMMAND STRING
               01960         ;
0127 3E40      01970 BUF     LD A,40H        ;IF ENTERED HERE ONLY UPPER CASE
0129 324640    01980         LD (UCFLAG),A   ;40H IS UPPER FLAG, 60H FOR LC
012C 23        01990 BUF0    INC HL          ;LEAVE ROOM FOR BUFFER COUNT
012D 0601      02000         LD B,01H        ;ROOM FOR COUNT,COMMA,AND 01H
012F CD5D02    02010 BUF1    CALL GBYTE      ;GET CHARACTER RETURN WITH DATA
0132 FE01      02020         CP 01H          ;ESCAPE FROM BUFFER ENTRY
0134 CA6600    02030         JP Z,RSTART     ;RETURN TO COMMAND, SP IS OK
0137 FE0D      02040         CP 0DH          ;CARRIAGE RETURN
0139 C8        02050         RET Z           ;RETURN TO CALLING ROUTINE
013A FE0C      02060         CP 0CH          ;CLEAR SCREEN
013C 2812      02070         JR Z,PCH        ;CHANGE DATA TO K COMMAND
013E FE08      02080 DELETE  CP 08H          ;BACKSPACE?
0140 2010      02090         JR NZ,PCHAR     ;NO INSERT CHAR INTO BUFFER
0142 3E01      02100 DLT     LD A,01H        ;BEGINNING?
0144 B8        02110         CP B            ;IS IT
0145 28E8      02120         JR Z,BUF1       ;CAN'T GO BACK FURTHER
0147 2B        02130 DLTE    DEC HL          ;DECREMENT BUFFER
0148 05        02140 DLTE1   DEC B           ;AND COUNT
0149 3E08      02150         LD A,08H
014B CDB201    02160         CALL CRT        ;REMOVE FROM SCREEN
014E 18DF      02170         JR BUF1         ;GET NEXT CHAR
0150 3E43      02180 PCH     LD A,'C'        ;CLEAR SCREEN
0152 04        02190 PCHAR   INC B           ;IS THERE ROOM?
0153 28F2      02200         JR Z,DLTE       ;NO SO DO AUTOMATIC 08H
0155 77        02210         LD (HL),A
0156 CDB201    02220         CALL CRT        ;LOAD IN BUFFER AND DISPLAY
0159 23        02230         INC HL          ;NEXT POSITION IN BUFFER
015A 18D3      02240         JR BUF1         ;GET NEXT CHARACTER
               02250         ;
               02260         ;FOLLOWING ARE PRINT ROUTINES AND DATA
               02270         ;CONDITIONING FOR THESE SUBROUTINES
               02280         ;
```

```
015C CDA201    02290 SPCHL   CALL SPACE       ;PRINT A SPACE FOLLOWED BY
015F 7E        02300 PPHL    LD A,(HL)        ;THE  NUMBER LOCATED AT (HL)
0160 CD6301    02310 PHEX    CALL PHEXH       ;TWO NIBBLES MUST BE OUTPUT
0163 0F        02320 PHEXH   RRCA             ;GET HIGH HEX VALUE
0164 0F        02330         RRCA
0165 0F        02340         RRCA
0166 0F        02350         RRCA             ;SWAP COMPLETE
0167 F5        02360 PHEXL   PUSH AF          ;GET LOW HEX VALUE
0168 E60F      02370         AND 0FH          ;MASK
016A FE0A      02380         CP 0AH           ;GREATER THAN 9
016C 3007      02390         JR NC,ADJUST     ;THEN NO CARRY
016E F630      02400         OR 30H           ;NOW ASCII FOR 0-9
0170 CDA401    02410 PHEX1   CALL PRNT        ;OUTPUT
0173 F1        02420         POP AF           ;GET AND ADJUST SP
0174 C9        02430         RET
0175 C637      02440 ADJUST  ADD A,37H        ;ADJUST FOR ALPHABETIC
0177 18F7      02450         JR PHEX1
               02460         ;
               02470         ;PRINTS NUMBER IN HL REGISTER AS ADDRESS
               02480         ;
0179 7C        02490 PNHL    LD A,H           ;PRINT H FIRST
017A CD6001    02500         CALL PHEX
017D 7D        02510         LD A,L           ;NOW  L
017E 18E0      02520         JR PHEX          ;RETURNS FROM CRT PROGRAM
0180 CD9E01    02530 ADDR    CALL NEWLIN      ;DO CR AND LINE FEED
0183 CD7901    02540 ADDR1   CALL PNHL        ;PRINTS HL WHICH IS ADDRESS
0186 3E3A      02550 ADDR2   LD A,':'         ;DELIMITER
0188 181A      02560         JR PRNT
               02570         ;
               02580         ;
               02590         ;PRINT MESSAGE
               02600         ;
018A 214300    02610 PMSG1   LD HL,SIGNON     ;FROLIC:
018D 7E        02620 PMSG    LD A,(HL)        ;GET DATA
018E FE01      02630         CP 01H           ;END OF MESSAGE MARK
0190 C8        02640         RET Z            ;RETURNS FROM MESSAGES VIA BREAK
0191 CDB201    02650         CALL CRT
0194 23        02660         INC HL           ;NEXT DATA
0195 18F6      02670         JR PMSG
               02680         ;
               02690         ;SHIFT DATA THROUGH HL PAIR
               02700         ;
0197 29        02710 LDHL    ADD HL,HL        ;SHIFT
0198 29        02720         ADD HL,HL
0199 29        02730         ADD HL,HL
019A 29        02740         ADD HL,HL
019B B5        02750         OR L             ;OR AC IN SHIFT PATTERN
019C 6F        02760         LD L,A           ;REPLACE L
019D C9        02770         RET
               02780         ;
               02790         ;PRINTS DATA IN A TO PRINT DEVICE
               02800         ;
019E 3E0D      02810 NEWLIN  LD A,0DH         ;LINE FEED
01A0 1802      02820         JR PRNT          ;WANT TO BREAK ON NEW LINES
01A2 3E20      02830 SPACE   LD A,20H         ;SPACE ADJUST
01A4 F5        02840 PRNT    PUSH AF          ;SAVE
01A5 CD6D02    02850 PRNT1   CALL CHKIN       ;HAS BREAK OR SPACE
```

```
01A8 FE01     02860          CP 01H           ;BEEN PRESSED
01AA CA6600   02870          JP Z,RSTART      ;BACK TO BUFFER
01AD FE20     02880          CP 20H
01AF 28F4     02890          JR Z,PRNT1       ;YES, WAIT FOR RELEASE
01B1 F1       02900          POP AF           ;FALL INTO CRT DISPLAY
              02910          ;
              02920          ;CRT DISPLAY
              02930          ;
01B2 D9       02940 CRT      EXX              ;CRT OUTPUT OF DATA IN A
01B3 F5       02950          PUSH AF          ;SAVE ON STACK
01B4 2A4040   02960          LD HL,(CURSOR)
01B7 3620     02970          LD (HL),20H      ;REMOVE CURSOR
01B9 FE0D     02980          CP 0DH           ;CARRIAGE RETURN
01BB 280A     02990          JR Z,LINEF
01BD FE08     03000          CP 08H           ;BACKSPACE
01BF 2818     03010          JR Z,BKSP
01C1 77       03020          LD (HL),A        ;PLACE CHARACTER ON SCREEN
01C2 23       03030          INC HL           ;ADVANCE CURSOR
01C3 CB74     03040          BIT 6,H          ;OFF PAGE?
01C5 2813     03050          JR Z,OUT2
01C7 01C003   03060 LINEF    LD BC,03C0H      ;1024-64 MEMORY BYTES
01CA 11003C   03070          LD DE,VIDEO
01CD 21403C   03080          LD HL,VIDEO+64D ;ONE LINE FROM PAGE TOP
01D0 EDB0     03090          LDIR
01D2 EB       03100          EX DE,HL         ;CLEAR LAST LINE
01D3 CDF201   03110          CALL CLEAR       ;CLEAR LAST LINE
01D6 21C13F   03120          LD HL,VIDEO+961D;BEGIN OF LAST LINE
01D9 2B       03130 BKSP     DEC HL
01DA 224040   03140 OUT2     LD (CURSOR),HL
01DD 3A4440   03150          LD A,(CMARK)     ;MONITOR OR USER?
01E0 77       03160          LD (HL),A        ;CURSOR MARK
01E1 F1       03170          POP AF           ;GET DATA FROM STACK
01E2 4F       03180          LD C,A           ;SAVE IN C
01E3 3A4A40   03190          LD A,(PRNTFG)    ;HARD COPY?
01E6 B7       03200          OR A
01E7 C41840   03210          CALL NZ,TTYP     ;IN RAM A JUMP TO USER ROUTINE
01EA 79       03220          LD A,C           ;RESTORE DATA IN C
01EB D9       03230          EXX              ;ADJUST THOSE REGISTERS
01EC C9       03240          RET
01ED 21003C   03250 CLEARS   LD HL,VIDEO
01F0 FD23     03260          INC IY           ;CORRECT BUFFER
01F2 CB74     03270 CLEAR    BIT 6,H          ;FINISHED
01F4 C0       03280          RET NZ           ;WHEN THIS BIT GOES ONE
01F5 3620     03290          LD (HL),20H      ;SPACE ON SCREEN
01F7 23       03300          INC HL           ;NEXT
01F8 18F8     03310          JR CLEAR         ;CONTINUE WITH CLEAR
              03320          ;
              03330          ;FILL ROUTINES FOR ADDRESS OR DATA
              03340          ;
01FA CD0302   03350 G3N      CALL G2N         ;GET TWO NUMBERS HL AND BC
01FD EB       03360          EX DE,HL         ;SAVE HL VALUE IN DE
01FE CD3202   03370          CALL GHL         ;GET 3RD ARGUMENT
0201 EB       03380          EX DE,HL         ;THIS GOES IN DE REGISTER
0202 C9       03390          RET              ;WITH HL START ADDRESS
0203 CD3202   03400 G2N      CALL GHL         ;GET 2 NUMBERS FIRST FOR HL
0206 EB       03410          EX DE,HL
0207 CDE502   03420          CALL GET1        ;GET LAST DATA IN BUFFER
```

```
020A 3821      03430          JR C,G2N3        ;FOR LONGEST INSTRUCTION
020C CCE302    03440 G2N0     CALL Z,GET       ;EITHER COULD BE T
020F 28FB      03450          JR Z,G2N0        ;SKIP SPACES
0211 FE54      03460          CP 'T'           ;THROUGH TILL LAST ADDRESS
0213 280D      03470          JR Z,G2N1        ;GO COMPUTE BLOCK LENGTH
0215 FD2B      03480          DEC IY           ;ADJUST POINTER
0217 CDE502    03490          CALL GET1        ;LAST IN BUFFER
021A C20003    03500          JP NZ,ERROR      ;ONLY VALID DELIMITERS
021D CD3202    03510          CALL GHL         ;GET BLOCK LENGTH
0220 1807      03520          JR G2N2          ;GO STORE IN HL
0222 CD3202    03530 G2N1     CALL GHL         ;END ADDRESS MUST COMPUTE
0225 3F        03540          CCF              ;CLEAR FLAG
0226 ED52      03550          SBC HL,DE
0228 23        03560          INC HL           ;NEED ONE MORE
0229 44        03570 G2N2     LD B,H           ;STORE LENGTH IN BC
022A 4D        03580          LD C,L
022B EB        03590          EX DE,HL         ;PUT START ADDRESS BACK
022C C9        03600          RET
022D 214000    03610 G2N3     LD HL,40H        ;MINIMUM LENGTH OF 40H
0230 18F7      03620          JR G2N2          ;STORE IN BC
               03630          ;
               03640          ;GET VALID HEX IN HL REGISTER
               03650          ;
0232 210000    03660 GHL      LD HL,0000H      ;INITIALIZE TO ZERO
0235 CDE302    03670 GHL0     CALL GET
0238 28FB      03680          JR Z,GHL0        ;SKIP SPACES
023A CD4C02    03690          CALL VALHEX      ;CHECK FOR VALIDITY
023D DA0003    03700          JP C,ERROR       ;THIS MUST CLEAR BREAK POINTS
0240 CD9701    03710 GHL1     CALL LDHL        ;SHIFT INTO HL PAIR
0243 CDE302    03720          CALL GET
0246 CD4C02    03730          CALL VALHEX
0249 30F5      03740          JR NC,GHL1
024B C9        03750          RET              ;AND RETURN TO CALLING PROGRAM
               03760          ;
               03770          ;
               03780          ;CHECK IF VALID HEX NO.
               03790          ;
024C C6D0      03800 VALHEX   ADD A,0D0H       ;IF BETWEEN 30 AND 39
024E 3F        03810          CCF              ;AND 41 AND 47 CARRY GENERATED
024F D8        03820          RET C            ;RETURN NOT VALID FLAG
0250 FE0A      03830          CP 0AH           ;IF LESS THAN 10 VALID
0252 3F        03840          CCF              ;VALID RETURN IS NO CARRY
0253 D0        03850          RET NC           ;BETWEEN 0 AND 9 AND VALID
0254 C6F9      03860          ADD A,0F9H       ;IF BETWEEN 17H AND 22H
0256 FE0A      03870          CP 0AH           ;THIS WILL FLAG
0258 D8        03880          RET C            ;ERROR THOSE FEW
0259 FE10      03890          CP 10H
025B 3F        03900          CCF              ;IF OVER 0FH NO CARRY
025C C9        03910          RET              ;RETURN WITH PROPER FLAG
               03920          ;
               03930          ;GET CHARACTERS
               03940          ;
025D E5        03950 GBYTE    PUSH HL
025E CD6D02    03960          CALL CHKIN
0261 214740    03970          LD HL,OLD
0264 BE        03980          CP (HL)
0265 28F7      03990          JR Z,GBYTE+1     ;WANT NEW CHARACTER
```

```
0267 77        04000          LD (HL),A
0268 B7        04010          OR A              ;NO NULLS
0269 28F3      04020          JR Z,GBYTE+1
026B E1        04030          POP HL
026C C9        04040          RET
026D 3A7F38    04050 CHKIN    LD A,(387FH)
0270 B7        04060          OR A
0271 C8        04070          RET Z
0272 D9        04080          EXX
0273 CD8302    04090 TWO      CALL KYD1         ;GET CHARACTER
0276 57        04100          LD D,A            ;SAVE
0277 0600      04110          LD B.00H          ;SHORT DELAY
0279 10FE      04120 D1       DJNZ D1
027B CD8302    04130          CALL KYD1
027E BA        04140          CP D              ;IS IT THE SAME
027F 20F2      04150          JR NZ,TWO         ;NOT SAME TRY AGAIN
0281 D9        04160          EXX
0282 C9        04170          RET
               04180          ;
               04190          ;KEY SCAN
               04200          ;
0283 010000    04210 KYD1     LD BC,0000H
0286 210138    04220          LD HL,3801H       :TURN ON BIT ZERO SCAN TO START
0289 7E        04230 NEXT     LD A,(HL)         ;GET DATA
028A B7        04240          OR A
028B 2008      04250          JR NZ,HDATA
028D CB25      04260          SLA L
028F 03        04270          INC BC
0290 F28902    04280          JP P,NEXT
0293 AF        04290          XOR A
0294 C9        04300          RET               ;ESCAPE
0295 1F        04310 HDATA    RRA
0296 3803      04320          JR C,LOWBIT
0298 04        04330          INC B
0299 18FA      04340          JR HDATA
029B CB75      04350 LOWBIT   BIT 6,L           ;LINE 6 SPECIAL CHARACTERS
029D 203C      04360          JR NZ,SPEC
029F 79        04370          LD A,C
02A0 07        04380          RLCA
02A1 07        04390          RLCA
02A2 07        04400          RLCA
02A3 B0        04410          OR B
02A4 4F        04420          LD C,A            ;5 LSB HAVE LOW DATA
02A5 3A8038    04430          LD A,(3880H)      ;IS SHIFT KEY PRESSED?
02A8 B7        04440          OR A              ;SHIFT KEY IN TWO PLACES
02A9 2802      04450          JR Z,NOSFT
02AB 3E01      04460          LD A,01H
02AD 47        04470 NOSFT    LD B,A            :INFORMATION OF SHIFT IN B
02AE 3E08      04480          LD A,08H
02B0 BD        04490          CP L              ;IF CARRY NOT ALPHA CHAR
02B1 380C      04500          JR C,NALPHA
02B3 3A4640    04510          LD A,(UCFLAG)     ;60H HERE IF LOWERCASE REQUIRED
               04520                            ;OTHERWISE 40H FOR ONLY UPPERCASE
02B6 B1        04530          OR  C
02B7 CB40      04540          BIT 0,B           ;WAS SHIFT PRESSED
02B9 2001      04550          JR NZ,CAPIT
02BB C9        04560          RET
```

```
02BC CBAF      04570 CAPIT   RES 5,A          ;AC HAS UPPER CASE DATA
02BE C9        04580         RET
02BF CB6D      04590 NALPHA  BIT 5,L
02C1 200B      04600         JR NZ,NOT7
02C3 3E30      04610 NALPH1  LD A,30H
02C5 B1        04620         OR C
02C6 CB40      04630         BIT 0,B
02C8 2001      04640         JR NZ,CHAR       ;THIS IS NOW CHARACTER
02CA C9        04650         RET
02CB CBA7      04660 CHAR    RES 4,A
02CD C9        04670         RET
02CE 3A2038    04680 NOT7    LD A,(3820H)
02D1 FE10      04690         CP 10H
02D3 38EE      04700         JR C,NALPH1
02D5 78        04710         LD A,B
02D6 EE01      04720         XOR 01H
02D8 47        04730         LD B,A
02D9 18E8      04740         JR NALPH1
02DB 213B00    04750 SPEC    LD HL,TABLE      :TABLE MUST BE LOCATED
02DE 78        04760         LD A,B           ;SUCH THAT THE FOLLOWING ADDITION
02DF 85        04770         ADD A,L          :DOES NOT CROSS BOUNDRIES
02E0 6F        04780         LD L,A
02E1 7E        04790         LD A,(HL)
02E2 C9        04800         RET
               04810         ;
               04820         ;GET FETCHES NEXT DATA IN INPUT STRING
               04830         :BY INC FIRST IY WILL ALWAYS POINT TO LAST VALUE
               04840         ;
02E3 FD23      04850 GET     INC IY
02E5 FD7E00    04860 GET1    LD A,(IY+0H)     ;ENTER HERE FOR LAST ENTRY
02E8 FE01      04870         CP 01H           ;BREAK?
02EA CA6C00    04880         JP Z,COMAND      ;START ALL OVER
02ED FE20      04890         CP 20H           ;SPACE SET ZERO FLAG
02EF C8        04900         RET Z            ;FLAG SPACES
02F0 FE2C      04910         CP ','           ;IF COMMA SET CARRY
02F2 C0        04920         RET NZ           ;ALL OTHERS RETURN
02F3 B7        04930 GET3    OR A             ;NO ZERO
02F4 37        04940         SCF              ;SET CARRY
02F5 D8        04950         RET C
               04960         ;
02F6 FD23      04970 CONTI1  INC IY           ;NEXT
02F8 CDE502    04980 CONTIN  CALL GET1        :LAST CHARACTER IN STRING
02FB 28F9      04990         JR Z,CONTI1      ;SKIP TRAILING SPACES
02FD DA8000    05000         JP C,CMND        ;FINISH OLD OR BACK TO BEGIN
0300 3E3F      05010 ERROR   LD A,'?'          :SHOW AT END
0302 FD7700    05020         LD (IY+0),A      ;SHOW IT
0305 CDE302    05030 ER0     CALL GET         ;ADJUST BUFFER
0308 3802      05040         JR C,ER1         ;WANT NEXT COMMA
030A 18F9      05050         JR ER0           ;OUT ONLY WITH TRUE COMMA
030C 218040    05060 ER1     LD HL,BUFFER     :0DH STARTS THIS LINE
030F CD8D01    05070         CALL PMSG        :OUTPUT BUFFER TEXT TO ERROR
0312 CD0701    05080         CALL CLRBP       ;CLEAR BREAKPOINTS
0315 3E00      05090 TOFF1   LD A,00H
0317 D3EC      05100 ER2     OUT (0ECH),A     ;ALL SET
0319 18DD      05110         JR CONTIN        ;CONTINUE WITH BUFFER
               05120         ;
               05130         ;COMMAND TABLE
```

```
                05140           ;
031B A503       05150 COMTAB    DEFW FILL       ;FILL ADDRESS SPACE WITH CONSTANT
031D 5103       05160           DEFW ASCII      ;ADDRESS OF ROUTINE
031F 6803       05170           DEFW BUFFR      ;ADDRESS OF BUFFER PROGRAM
0321 ED01       05180           DEFW CLEARS     ;CLEAR SCREEN WITH 20H
0323 9203       05190           DEFW DISPLY     ;ADDRESS OF ROUTINE
0325 C703       05200           DEFW GO         ;EXECUTE WITH CURRENT SCREEN
0327 C307       05210           DEFW FINDON     ;SEARCH FOR SINGLE BYTE
0329 B403       05220           DEFW ENTER      ;EXECUTE WITH OLD SCREEN DATA
032B 6D04       05230           DEFW HEXM       ;ADDRESS OF HEXM
032D C404       05240           DEFW SUBSM      ;INSERT FROM BUFFER
032F 0003       05250           DEFW ERROR      ;J
0331 0003       05260           DEFW ERROR      ;K
0333 A104       05270           DEFW LIST       ;TOGGLE HARDCOPY SWITCH
0335 AC04       05280           DEFW MOVE       ;ADDRESS OF MOVE
0337 8204       05290           DEFW IN         ;INPUT FROM PORT
0339 BA04       05300           DEFW OUT        ;OUTPUT TO PORT
033B 1B40       05310           DEFW PROM       ;PROM PROGRAMMER
033D BF07       05320           DEFW FINDTO     ;SEARCH FOR TWO BYTE INTEL FORMAT
033F 1107       05330           DEFW READ       ;READ A CASSETTE TAPE WITH BIAS
0341 E704       05340           DEFW INSERT     ;DATA FROM KEYBOARD
0343 B403       05350           DEFW ENTER      ;TRANSFER FROM SCREEN TO VIDEO
0345 1E40       05360           DEFW USER       ;USER ROUTINE
0347 3405       05370           DEFW VERIFY     ;ADDRESS OF VERIFY
0349 5906       05380           DEFW WRIT       ;WRITE A TAPE ON CASSETTE
034B 5305       05390           DEFW EXAM       ;ADDRESS
034D 0003       05400           DEFW ERROR
034F AF03       05410           DEFW STEP       ;SINGLE STEP IF HARDWARE ADDED
                05420                           ;WITH OLD SCREEN DATA
                05430           ;
                05440           ;COMMAND SUBROUTINES
                05450           ;
                05460           ;DISPLAY IN ASCII FORMAT WITH GRAPHICS
                05470           ;
0351 CD0302     05480 ASCII     CALL G2N        ;GET ARGUMENTS
0354 1620       05490 ASCII1    LD D,32D        ;56 DECIMAL POSITIONS MAX
0356 CD8001     05500           CALL ADDR       ;PRINT LOCATION OF BEGINNING LINE
0359 CDA201     05510           CALL SPACE      ;SPACE
035C 7E         05520 ASCII2    LD A,(HL)       ;GET DATA
035D CDB201     05530 ASCII4    CALL CRT
0360 EDA1       05540           CPI             ;ASCII DUMP FINISHED?
0362 E0         05550           RET PO          ;CHANGE LATER FOR STRING
0363 15         05560           DEC D           ;FINISHED WITH LINE?
0364 28EE       05570           JR Z,ASCII1     ;NOT FINISHED WITH LINE
0366 18F4       05580           JR ASCII2       ;NOT FINISHED
                05590           ;
                05600           ;BUFFER SAVE AND EXECUTE ROUTINES
                05610           ;
0368 CDE302     05620 BUFFR     CALL GET        ;GET NEXT DATA
036B DA8103     05630           JP C,TRAN       ;TRANSFER SAVED BUFFER
036E FE53       05640           CP 'S'          ;SAVE BUFFER
0370 2801       05650           JR Z,SAVE       ;IGNORE ALL ELSE
0372 C9         05660           RET             ;CONTINUE AND IGNORE
0373 218040     05670 SAVE      LD HL,BUFFER    ;BUFFER BEGINNING
0376 010001     05680           LD BC,100H      ;PLENTY
0379 118041     05690           LD DE,TEM       ;PLACE TO SAVE
037C EDB0       05700           LDIR            ;TRANSFER COMMAND TEXT
```

```
037E FD23      05710          INC IY           ;BYPASS THE S FOR CONTIN
0380 C9        05720          RET
0381 218041    05730 TRAN     LD HL,TEM
0384 010001    05740          LD BC,100H       ;ONE PAGE OF BUFFER
0387 118040    05750          LD DE,BUFFER
038A D5        05760          PUSH DE
038B FDE1      05770          POP IY
038D EDB0      05780          LDIR
038F C38A00    05790          JP CD            ;QUICK WAY TO LOAD BUFFER
               05800          ;
               05810          ;
               05820          ;DISPLAY IN HEX FORMAT
               05830          ;
0392 CD0302    05840 DISPLY   CALL G2N          ;GET ARGUMENTS
0395 CD8001    05850 DISP1    CALL ADDR         ;PRINT MEM ADDRESS
0398 CD5C01    05860 DISP2    CALL SPCHL        ;SPACE FOLLOWED BY HEX DATA
039B EDA1      05870          CPI              ;INCREMENT POINTERS
039D E0        05880          RET PO           ;BC=0 THEN DONE
039E 7D        05890          LD A,L           ;GET ADDRESS OF MEM
039F E60F      05900          AND 0FH          ;MASK ALL BUT LOW
03A1 28F2      05910          JR Z,DISP1       ;NEXT LINE
03A3 18F3      05920          JR DISP2         ;GO FINISH LINE
               05930          ;
               05940          ;
               05950          ;
               05960          ;FILL MEMORY WITH CONSTANT
               05970          ;
03A5 CDFA01    05980 FILL     CALL G3N         ;GET 3ARGUMENTS, WHERE,
03A8 7B        05990          LD A,E           ;HOW MANY AND WHAT
03A9 77        06000 FILL1    LD (HL),A        ;DO IT
03AA EDA1      06010          CPI              ;INCREMENT POINTERS
03AC E0        06020          RET PO           ;FINISHED?
03AD 18FA      06030          JR FILL1         ;CONTINUE
               06040          ;
               06050          ;
               06060          ;SINGLE STEP EXECUTE WITH LAST SCREEN DATA
03AF 3E01      06070 STEP     LD A,01          ;SINGLE STEP FLAG
03B1 324240    06080          LD (SSFLAG),A    ;STORE AND FALL INTO GO
               06090          ;
03B4 2A3E40    06100 ENTER    LD HL,(SCREEN)   ;CURRENT SCREEN STORAGE
03B7 11003C    06110          LD DE,3C00H      ;TO BE PLACED HERE
03BA 010004    06120          LD BC,0400H      ;ALL 1024 OF THEM
03BD EDB0      06130          LDIR             ;DOES IT
03BF CDE502    06140          CALL GET1        ;LAST IN STRING
03C2 FE54      06150          CP 'T'           ;TRANSFER IS ALL THAT IS
03C4 CA6F00    06160          JP Z,COM1        ;REQUIRED THEN RETURN NO PROMPT
               06170          ;
               06180          ;GET PC AND EXECUTE WITHOUT SCREEN
               06190          ;IF ENTERED AT GO
               06200          ;
               06210          ;
03C7 0600      06220 GO       LD B,00H         ;BREAK COUNTER
03C9 FD23      06230          INC IY           ;ADJUST BUFFER
03CB 211240    06240          LD HL,RSTLOC
03CE 36C3      06250          LD (HL),0C3H     ;PUT JUMP THERE
03D0 21A600    06260          LD HL,RSAVE      ;ADDRESS FOR JUMP
03D3 221340    06270          LD (RSTLOC+1),HL;ADDRESS
```

```
03D6 318040    06280          LD SP,BREAKS     ;BREAKSTACK
03D9 CDE502    06290 GO1      CALL GET1        ;NEXT DATA IN BUFFER
03DC CCE302    06300 GO2      CALL Z,GET       ;NEXT IN BUFFER
03DF 28FB      06310          JR Z,GO2
03E1 3841      06320          JR C,LDREG       ;USE STORED PC
03E3 FE2E      06330          CP '.'           ;BREAKPOINT?
03E5 280A      06340          JR Z,BREAK       ;IT IS A BREAK
03E7 FD2B      06350          DEC IY           ;ADJUST FOR 1ST DIGIT OF ADDR
03E9 CD3202    06360          CALL GHL         ;START OR BREAK
03EC 223C40    06370          LD (PCSAVE),HL   ;NEW PC
03EF 18E8      06380          JR GO1           ;CONTINUE
03F1 CDE302    06390 BREAK    CALL GET         ;NEXT CHARACTER
03F4 FE2B      06400          CP '+'           ;RELATIVE JUMP
03F6 2816      06410          JR Z,PLUS
03F8 FE2D      06420          CP '-'
03FA 281C      06430          JR Z,MINUS
03FC FD2B      06440          DEC IY           ;NO RELATIVE JUMP
03FE CD3202    06450          CALL GHL         ;GET ABSOLUTE VALUE
0401 7E        06460 BREAK1   LD A,(HL)        ;GET CODE AT ADDRESS
0402 E5        06470          PUSH HL          ;SAVE ADDRESS
0403 F5        06480          PUSH AF          ;SAVE DATA
0404 3EFF      06490          LD A,RST         ;RESTART ,TRS-80 USES 7 FOR TIME
0406 77        06500          LD (HL),A        ;SAVE IT AS BREAK
0407 04        06510          INC B            ;BREAK COUNTER
0408 78        06520          LD A,B
0409 324540    06530          LD (BPNO),A      ;SAVE
040C 18CB      06540          JR GO1           ;CONTINUE
040E CD3202    06550 PLUS     CALL GHL         ;GET RELATIVE VALUE
0411 EB        06560          EX DE,HL         ;SAVE IN DE
0412 2A3C40    06570          LD HL,(PCSAVE)   ;PROGRAM COUNTER
0415 19        06580          ADD HL,DE        ;BREAKPOINT COMPUTED
0416 18E9      06590          JR BREAK1
0418 CD3202    06600 MINUS    CALL GHL         ;GET RELATIVE VALUE
041B EB        06610          EX DE,HL
041C 2A3C40    06620          LD HL,(PCSAVE)
041F B7        06630          OR A             ;CLEAR CARRY
0420 ED52      06640          SBC HL,DE
0422 18DD      06650          JR BREAK1        ;RELATIVE JUMP COMPUTED
0424 FD224840  06660 LDREG    LD (EPOINT),IY   ;SAVE BUFFER POINTER AT BPOINT
0428 21FBD3    06670          LD HL,0D3FBH     ;ENABLE INTERRUPT AND OUT
042B 22A042    06680          LD (JMP),HL
042E 2100C3    06690          LD HL,0C300H     ;PORT AND JUMP
0431 22A242    06700          LD (JMP+2),HL
0434 2A3C40    06710          LD HL,(PCSAVE)   ;EXECUTE ADDRESS
0437 22A442    06720          LD (JMP+4),HL    ;ADDRESS IN PLACE
043A 312640    06730          LD SP,BOTMS      ;REGISTER DATA
043D DDE1      06740          POP IX
043F FDE1      06750          POP IY           ;RESTORE IX AND IY
0441 F1        06760          POP AF           ;INTERRUPT INFORMATION
0442 ED47      06770          LD I,A           ;LOAD INTERRUPT VECTOR
0444 3001      06780          JR NC,GO4        ;IF CARRY NO INTERRUPTS
0446 FB        06790          EI               ;INTERRUPTS REQUESTED
0447 F1        06800 GO4      POP AF           ;THIS TIME IT IS AF'
0448 C1        06810          POP BC
0449 D1        06820          POP DE
044A E1        06830          POP HL           ;PRIMES IN PLACE
044B D9        06840          EXX
```

```
044C 08        06850          EX AF,AF'          ;NOW SWITCH
044D F1        06860          POP AF
044E C1        06870          POP BC
044F D1        06880          POP DE
0450 2A2440    06890          LD HL,(SSTACK)     ;GET SYSTEM STACK POINTER
0453 F9        06900          LD SP,HL           ;STACK IN PLACE
0454 2A3A40    06910          LD HL,(HLSAVE)     ;HL OK
0457 F5        06920          PUSH AF            ;SAVE AF ON USER STACK
0458 3A4240    06930          LD A,(SSFLAG)      ;SINGLE STEP?
045B B7        06940          OR A               ;TEST
045C 280B      06950          JR Z,POPAF         ;IF ZERO NO SINGLE STEP
045E 3E08      06960          LD A,08H           ;ENABLE INTERRUPT FROM EXTERNAL BUS
0460 D3E0      06970          OUT (0E0H),A       ;USING PORT E0H
0462 07        06980          RLCA               ;08 BECOMES 10 WHICH ENABLES
0463 D3EC      06990          OUT (0ECH),A       ;ADDRESS, DATA, AND CONTROL LINES
               07000          ;
               07010          ;WITH HARDWARE TO COUNT OUT TO PRODUCE INTERRUPT
               07020          ;
0465 F1        07030          POP AF             ;RESTORE REGISTERS
0466 C3A042    07040          JP JMP             ;SET SHIFT REGISTER FOR SINGLE STEP
0469 F1        07050 POPAF    POP AF             ;RESTORE REGISTERS
046A C3A342    07060          JP JMP+3           ;EXECUTE BYPASSING STEP INTERRUPT
               07070          ;
               07080          ;HEX MATH FIRST SUM THEN DIFFERENCE
               07090          ;
046D CD3202    07100 HEXM     CALL GHL           ;GET ARGUMENTS
0470 EB        07110          EX DE,HL
0471 CD3202    07120          CALL GHL           ;2ND ARGUMENT
0474 E5        07130          PUSH HL            ;SAVE 2ND ARGUMENT
0475 19        07140          ADD HL,DE          ;GET SUM
0476 CD8301    07150          CALL ADDR1         ;NUMBER FOLLOWED BY :
0479 E1        07160          POP HL             ;GET 2ND BACK
047A EB        07170          EX DE,HL           ;PUT IN PROPER ORDER
047B AF        07180          XOR A              ;CLEAR BORROW FLAG
047C ED52      07190          SBC HL,DE          ;GET DIFFERENCE
047E CD7901    07200          CALL PNHL          ;DISPLAY SECOND NUMBER
0481 C9        07210          RET                ;FINISHED
               07220          ;
               07230          ;INPUT FROM PORT XX AND DISPLAY
               07240          ;
0482 CD0302    07250 IN       CALL G2N           ;GET PORT DESIRED
0485 41        07260          LD B,C             ;NUMBER OF INPUT PORTS IN SEQUENCE
0486 4D        07270          LD C,L             ;C HAS PORT TO INPUT
0487 79        07280 INNEXT   LD A,C             ;A GETS PORT NUMBER
0488 CD6001    07290          CALL PHEX          ;OUTPUT PORT NUMBER
048B CD8601    07300          CALL ADDR2         ;OUTPUT A :
048E CDA201    07310          CALL SPACE
0491 ED78      07320          IN A,(C)           ;GET DATA
0493 CD6001    07330          CALL PHEX          ;OUTPUT DATA
0496 CDA201    07340          CALL SPACE         ;SPACE
0499 CDA201    07350          CALL SPACE
049C 05        07360          DEC B              ;ALL PORTS INPUT
049D C8        07370          RET Z              ;YES COMMAND STRING
049E 0C        07380          INC C              ;NEXT PORT
049F 18E6      07390          JR INNEXT
               07400          ;
04A1 FD23      07410 LIST     INC IY             ;ADJUST POINTER
```

```
04A3 3A4A40    07420          LD A,(PRNTFG)    ;GET CURRENT FLAG
04A6 EE01      07430          XOR 01H          ;TOGGLE TO OPPOSITE STATE
04A8 324A40    07440          LD (PRNTFG),A    ;RETURN TO STORAGE
04AB C9        07450          RET              ;CONTINUE STRING
               07460          ;
               07470          ;MOVE MEMORY
               07480          ;
04AC CDFA01    07490 MOVE     CALL G3N         ;GET ARGUMENTS OF FROM
04AF E5        07500 MOV1     PUSH HL          ;HOW MUCH AND TO
04B0 D5        07510          PUSH DE
04B1 C5        07520          PUSH BC          ;SAVE ARGUMENTS FOR VERIFY
04B2 EDB0      07530          LDIR             ;ONE OF THOSE NICE Z-80 CODES
04B4 C1        07540          POP BC           ;RESTORE
04B5 D1        07550          POP DE
04B6 E1        07560          POP HL           ;RESTORED NOW
04B7 C33705    07570          JP VERIF1        ;JUMP TO VERIFY
               07580          ;
               07590          ;OUTPUT HEX DATA TO PORT XX
               07600          ;
04BA CD3202    07610 OUT      CALL GHL         ;GET PORT NUMBER
04BD 4D        07620          LD C,L           ;PORT MUST BE IN C
04BE CD3202    07630          CALL GHL         ;GET DATA
04C1 ED69      07640          OUT (C),L        ;OUTPUT
04C3 C9        07650          RET              ;FINISHED
               07660          ;
               07670          ;SUBSTITUTE MEMORY FROM STRING OR KEYBOARD
               07680          ;
04C4 CD3202    07690 SUBSM    CALL GHL         ;GET ADDRESS
04C7 CD8001    07700 SUBS     CALL ADDR        ;PRINT IT
04CA E5        07710 SUBS1    PUSH HL          ;SAVE IT
04CB CD3202    07720          CALL GHL         ;GET NUMBER IN HL
04CE CDE502    07730          CALL GET1        ;LAST IN STRING
04D1 FE2F      07740          CP '/'           ;ADDRESS OR DATA
04D3 280D      07750          JR Z,NEWAD1      ;IT IS ADJUST STACK AND IY
04D5 7D        07760 SUBS3    LD A,L           ;ONLY L USED
04D6 E1        07770          POP HL           ;GET ADDRESS
04D7 77        07780          LD (HL),A        ;STORE
04D8 CD5C01    07790          CALL SPCHL       ;SHOW US
04DB 23        07800          INC HL           ;NEXT FORWARD POSITION
04DC CDE502    07810          CALL GET1        ;LAST STRING ENTRY
04DF D8        07820          RET C            ;WAY OUT
04E0 18E8      07830          JR SUBS1         ;CONTINUE INSERTION OF DATA
04E2 FD23      07840 NEWAD1   INC IY           ;ADJUST IY
04E4 D1        07850 NEWADR   POP DE           ;DON'T DESTROY NEW ADDRESS
04E5 18E0      07860          JR SUBS          ;PRINT NEW ADDRESS
04E7 CD3202    07870 INSERT   CALL GHL         ;GET ADDRESS
04EA CD8001    07880 INS1     CALL ADDR
04ED CD5C01    07890 INS2     CALL SPCHL       ;SPACE FOLLOWED BY DATA
04F0 E5        07900          PUSH HL          ;SAVE ADDRESS
04F1 210000    07910          LD HL,00H        ;CLEAR DATA
04F4 0601      07920          LD B,01H         ;COUNT
04F6 3E3F      07930 PRINIT   LD A,'?'         ;PROMPT FOR SUBSTITUTE KEYBOARD
04F8 CDB201    07940          CALL CRT         ;SHOW WE ARE LOOKING FOR DATA
04FB CD5D02    07950 INS3     CALL GBYTE       ;DATA FROM KEYBOARD
04FE FE0D      07960          CP 0DH           ;ENTER?
0500 282E      07970          JR Z,FIRST
0502 FE08      07980          CP 08H           ;GO  BACK
```

```
0504 2826      07990          JR Z,BACKW       ;RETURN PRINT ADDRESS
0506 FE20      08000          CP 20H           ;WILL NOT ALTER DATA
0508 2813      08010          JR Z,INCK        ;INCREMENT AND KEEP
050A CDB201    08020          CALL CRT
050D FE2F      08030          CP '/'           ;NEW ADDRESS
050F 2818      08040          JR Z,NADDS       ;LAST STRING WAS ADDRESS
0511 CD4C02    08050          CALL VALHEX      ;IS IT HEX
0514 38E0      08060          JR C,PRINIT      ;TRY AGAIN
0516 04        08070          INC B
0517 CD9701    08080          CALL LDHL        ;SHIFT INTO HL PAIR
051A C3FB04    08090          JP INS3
051D E1        08100 INCK     POP HL           ;GET OLD ADDRESS
051E 1803      08110          JR ENT1
0520 7D        08120 ENT      LD A,L           ;ONLY INFO IN L REQUIRED
0521 E1        08130          POP HL
0522 77        08140          LD (HL),A
0523 CD5C01    08150 ENT1     CALL SPCHL       ;SPACE FOLLOWED BY DATA
0526 23        08160          INC HL
0527 18C1      08170          JR INS1
0529 D1        08180 NADDS    POP DE
052A 18BE      08190          JR INS1
052C E1        08200 BACKW    POP HL
052D 2B        08210          DEC HL
052E 18BA      08220          JR INS1          ;PRINT LAST ADDRESS
0530 10EE      08230 FIRST    DJNZ ENT
0532 E1        08240          POP HL
0533 C9        08250          RET
               08260          ;
               08270          ;VERIFY MEMORY
               08280          ;
0534 CDFA01    08290 VERIFY   CALL G3N         ;GET ARGUMENTS
0537 1A        08300 VERIF1   LD A,(DE)
0538 EDA1      08310          CPI
053A 2813      08320          JR Z,VERF2       ;AGREES SO CONTINUE
053C 2B        08330          DEC HL           ;GO BACK FOR DISPLAY
053D CD8001    08340          CALL ADDR        ;ADDRESS
0540 CD5C01    08350          CALL SPCHL       ;AND DATA AT ADDRESS
0543 EB        08360          EX DE,HL         ;NOW OTHER
0544 CDA201    08370          CALL SPACE       ;SPACE
0547 CD8301    08380          CALL ADDR1       ;ADDRESS BUT NOT NEW LINE
054A CD5C01    08390          CALL SPCHL       ;DATA AT OTHER ADDRESS
054D EB        08400          EX DE,HL         ;RESTORE
054E 23        08410          INC HL           ;RESTORE HL
054F 13        08420 VERF2    INC DE           ;INC OTHER POINTER
0550 E0        08430          RET PO           ;FINISHED?
0551 18E4      08440          JR VERIF1        ;CONTINUE
               08450          ;
               08460          ;
0553 CDE302    08470 EXAM     CALL GET         ;NEXT IN STRING
0556 28FB      08480          JR Z,EXAM
0558 3821      08490          JR C,XALL        ;IF C, EXAMINE ALL
055A F5        08500          PUSH AF          ;SAVE
055B CDE302    08510          CALL GET         ;PRIME OR NOT
055E D627      08520          SUB '''          ;IF PRIME STORE 01H IN PRIMF
0560 2804      08530          JR Z,EXAM2       ;NO NEED TO CORRECT POINTER
0562 FD2B      08540          DEC IY           ;ADJUST BUFFER POINTER
0564 3EFF      08550          LD A,0FFH        ;WHEN INC WILL BE 0
```

```
0566 3C       08560 EXAM2   INC A             ;IT IS NOW 01H
0567 324B40   08570         LD (PRIMF),A      ;FLAG REFLECTS PRIME REGISTER
056A F1       08580 EXAM3   POP AF            :GET REGISTER NAME
056B 213906   08590         LD HL,XTAB
056E 012000   08600         LD BC,32D         ;32 CHARACTERS AND DATA LOCATIONS
0571 EDA1     08610 CONTCP  CPI               :COMPARE (HL) TO A AND INC
0573 CA0606   08620         JP Z,FOUND
0576 E20003   08630         JP PO,ERROR       ;NOT IN TABLE
0579 18F6     08640         JR CONTCP         ;CONTINUE COMPARISON
057B AF       08650 XALL    XOR A             ;CLEAR A
057C 324B40   08660         LD (PRIMF),A      ;MUST BE ZERO
057F 215706   08670         LD HL,PCP         ;PROGRAM COUNTER IN XTAB
0582 CDBB05   08680         CALL PVAL         ;PRINT 'P' AND CONTENTS OF PC
0585 CDA201   08690         CALL SPACE
0588 213906   08700         LD HL,XTAB        ;REGISTER TABLE
058B 0606     08710         LD B,6H           ;REST OF REGISTERS
058D CDAE05   08720         CALL ALL1         ;DISPLAY S,IX,IY,N,I,V
0590 0609     08730 ALL3    LD B,9H           :NINE REGISTER DATA DISPLAY
0592 CD9E01   08740         CALL NEWLIN
0595 CDAE05   08750         CALL ALL1
0598 E5       08760         PUSH HL
0599 C5       08770         PUSH BC
059A CDEF05   08780         CALL FLAGP        ;PRINT FLAG DATA
059D C1       08790         POP BC
059E E1       08800         POP HL
059F 3A4B40   08810         LD A,(PRIMF)
05A2 3C       08820         INC A
05A3 324B40   08830         LD (PRIMF),A
05A6 FE02     08840         CP 02H
05A8 C8       08850         RET Z
05A9 214506   08860         LD HL,RA
05AC 18E2     08870         JR ALL3
05AE C5       08880 ALL1    PUSH BC
05AF CDBB05   08890         CALL PVAL         ;PRINT NAME AND VALUE
05B2 F5       08900         PUSH AF
05B3 CDA201   08910         CALL SPACE
05B6 F1       08920         POP AF
05B7 C1       08930         POP BC
05B8 10F4     08940         DJNZ ALL1         ;REST OF REGISTERS
05BA C9       08950         RET
05BB 46       08960 PVAL    LD B,(HL)         ;HL POINTS TO REG NAME
05BC 23       08970         INC HL            :POINT TO BIAS
05BD 7E       08980         LD A,(HL)         ;SIGN WILL HAVE DATA AS TO 1 OR 2
05BE 112440   08990 PVAL1   LD DE,SSTACK
05C1 83       09000         ADD A,E
05C2 5F       09010         LD E,A
05C3 7A       09020         LD A,D            ;GET HIGH ORDER
05C4 CE00     09030         ADC A,00H         ;PAGE BOUNDARY
05C6 57       09040         LD D,A            :DOUBLE BYTE ADD
05C7 23       09050         INC HL            ;POINT TO NEXT ENTRY
05C8 78       09060         LD A,B            ;PRINT NAME
05C9 CDB201   09070         CALL CRT
05CC 3A4B40   09080         LD A,(PRIMF)
05CF B7       09090         OR A
05D0 2808     09100         JR Z,NPRIM
05D2 7B       09110         LD A,E
05D3 D608     09120         SUB 8H
```

```
05D5 5F       09130          LD E,A
05D6 3E27     09140          LD A,''''
05D8 1802     09150          JR PRIME
05DA 3E3D     09160 NPRIM    LD A,'='
05DC CDB201   09170 PRIME    CALL CRT
05DF CB7B     09180          BIT 7,E          ;CHECK FOR ONE OR TWO BYTE NUMBER
05E1 2807     09190          JR Z,ONEB        ;NO ADJUST IF ONE BYTE
05E3 CBBB     09200          RES 7,E          ;TWO BYTE, CORRECT BIAS
05E5 1A       09210          LD A,(DE)        ;GET VALUE
05E6 CD6001   09220          CALL PHEX        ;PRINT IT
05E9 1B       09230          DEC DE           ;DECREMENT POINTER
05EA 1A       09240 ONEB     LD A,(DE)        ;GET VALUE
05EB CD6001   09250          CALL PHEX
05EE C9       09260          RET
05EF 47       09270 FLAGP    LD B,A
05F0 CDA201   09280          CALL SPACE
05F3 213106   09290          LD HL,FTAB
05F6 CB20     09300 FLAG1    SLA B
05F8 3805     09310          JR C,OFLAG
05FA C8       09320          RET Z
05FB 3E20     09330          LD A,20H
05FD 1801     09340          JR FLAG2
05FF 7E       09350 OFLAG    LD A,(HL)
0600 CDB201   09360 FLAG2    CALL CRT
0603 23       09370          INC HL
0604 18F0     09380          JR FLAG1
0606 4E       09390 FOUND    LD C,(HL)        ;SAVE BIAS AND NUMBER OF BYTES
0607 47       09400          LD B,A           ;B REGISTER GETS NAME
0608 79       09410          LD A,C           ;BIAS INTO AC
0609 CDBE05   09420          CALL PVAL1       ;WRITE FORMER CONTENTS
060C CDE302   09430 FOUND1   CALL GET         ;DATA TO REPLACE OR END
060F 3811     09440          JR C,NOTTWO      ;PRINT NEW DATA
0611 28F9     09450          JR Z,FOUND1      ;SKIP BLANK (20H)
0613 FD2B     09460          DEC IY           ;ADJUST BUFFER POINTER GHL WANTS
0615 CD3202   09470          CALL GHL         ;VALID HEX IN FIRST POSITION
0618 7D       09480          LD A,L           ;MOVE TO STACK
0619 12       09490          LD (DE),A        ;IF ONLY ONE ITS IN L
061A 79       09500          LD A,C           ;SIGN HAS 1,2 INFORMATION
061B B7       09510          OR A             ;SET FLAG
061C F22206   09520          JP P,NOTTWO      ;IF POSITIVE ONLY ONE BYTE
061F 13       09530          INC DE
0620 7C       09540          LD A,H           ;GET HIGH DATA
0621 12       09550          LD (DE),A
0622 CDA201   09560 NOTTWO   CALL SPACE
0625 79       09570          LD A,C           ;GET REGISTER INFORMATION
0626 CDBE05   09580          CALL PVAL1
0629 4F       09590          LD C,A           ;DATA LAST PRINTED INTO C
062A 78       09600          LD A,B           ;CHECK FOR FLAG
062B FE46     09610          CP 'F'           ;IF FLAG PRINT FLAGS
062D 79       09620          LD A,C           ;DATA BACK TO A FOR FLAG PRINT
062E 28BF     09630          JR Z,FLAGP
0630 C9       09640          RET              ;RETURN TO STRING PROCESS
0631 53       09650 FTAB     DEFM 'SZXHX/NC'
0639 53       09660 XTAB     DEFM 'S'
063A 81       09670          DEFB 81H
063B 58       09680          DEFM 'X'
063C 83       09690          DEFB 83H
```

```
063D 59          09700          DEFM 'Y'
063E 85          09710          DEFB 85H
063F 49          09720          DEFM 'I'
0640 07          09730          DEFB 07H
0641 4E          09740          DEFM 'N'
0642 06          09750          DEFB 06H
0643 56          09760          DEFM 'V'
0644 9B          09770          DEFB 9BH
0645 41          09780 RA       DEFM 'A'
0646 11          09790          DEFB 11H
0647 42          09800          DEFM 'B'
0648 13          09810          DEFB 13H
0649 43          09820          DEFM 'C'
064A 12          09830          DEFB 12H
064B 44          09840          DEFM 'D'
064C 15          09850          DEFB 15H
064D 45          09860          DEFM 'E'
064E 14          09870          DEFB 14H
064F 48          09880          DEFM 'H'
0650 17          09890          DEFB 17H
0651 4C          09900          DEFM 'L'
0652 16          09910          DEFB 16H
0653 4D          09920          DEFM 'M'
0654 97          09930          DEFB 97H
0655 46          09940          DEFM 'F'
0656 10          09950          DEFB 10H
0657 50          09960 PCP      DEFM 'P'
0658 99          09970          DEFB 99H
                 09980 ;
                 09990 ;        WRITE TO CASSETTE TAPE
                 10000 ;
0659 CDFA01      10010 WRIT     CALL G3N
065C D5          10020          PUSH DE
065D E5          10030          PUSH HL
065E C5          10040          PUSH BC
065F 3E02        10050 GNAM3    LD A,2H
0661 D3EC        10060          OUT (0ECH),A
0663 AF          10070          XOR A           ;ZERO ACCUMULATOR
0664 CDDD06      10080 LEAD     CALL TOUT
0667 10FB        10090          DJNZ LEAD
0669 3EA5        10100 WRIT1    LD A,0A5H       ;SYNC MARK
066B CDDD06      10110          CALL TOUT
066E 3E55        10120          LD A,55H        ;FILE NAME MARK
0670 CDDD06      10130          CALL TOUT       ;WRITE NAME HEADER
0673 0606        10140          LD B,06H        ;6 BYTE NAME
0675 CDE502      10150          CALL GET1       ;LAST ENTRY
0678 380E        10160          JR C,FINB       ;IF COMMA NO NAME FILL BLANKS
067A CDE302      10170 WNAME    CALL GET        ;NEXT ENTRY
067D 3809        10180          JR C,FINB       ;FINISH WITH BLANKS
067F CDDD06      10190          CALL TOUT       ;WRITE NAME
0682 10F6        10200          DJNZ WNAME
0684 FD23        10210          INC IY          ;ADJUST BUFFER
0686 1807        10220          JR WRIT3        ;FULL NAME
0688 3E20        10230 FINB     LD A,20H        ;FINISH WITH BLANKS
068A CDDD06      10240          CALL TOUT
068D 10F9        10250          DJNZ FINB
068F D1          10260 WRIT3    POP DE
```

```
0690 E1        10270          POP HL
0691 15        10280 WRIT2    DEC D
0692 FAC106    10290          JP M,FINISH
0695 AF        10300          XOR A
0696 CD9B06    10310          CALL TDATA
0699 18F6      10320          JR WRIT2
069B 47        10330 TDATA    LD B,A
069C 3E3C      10340          LD A,3CH
069E CDDD06    10350          CALL TOUT
06A1 78        10360          LD A,B
06A2 CDDD06    10370          CALL TOUT
06A5 7D        10380          LD A,L
06A6 CDDD06    10390          CALL TOUT
06A9 7C        10400          LD A,H
06AA CDDD06    10410          CALL TOUT
06AD 7C        10420          LD A,H
06AE 22FE3F    10430          LD (3FFEH),HL   ;SHOW SOMETHING
06B1 7C        10440          LD A,H
06B2 85        10450          ADD A,L         ;START OF CHECKSUM
06B3 4F        10460          LD C,A          ;CHECKSUM STORAGE
06B4 7E        10470 TDATA1   LD A,(HL)
06B5 CDDD06    10480          CALL TOUT
06B8 7E        10490          LD A,(HL)
06B9 81        10500          ADD A,C
06BA 4F        10510          LD C,A
06BB 23        10520          INC HL
06BC 10F6      10530          DJNZ TDATA1
06BE 79        10540          LD A,C
06BF 181C      10550          JR TOUT
06C1 AF        10560 FINISH   XOR A
06C2 BB        10570          CP E
06C3 2804      10580          JR Z,WRIT5      ;NOTHING TO WRITE EXCEPT HEAD
06C5 7B        10590          LD A,E
06C6 CD9B06    10600          CALL TDATA
06C9 3E78      10610 WRIT5    LD A,78H        ;LAST BLOCK HEADER
06CB CDDD06    10620          CALL TOUT
06CE E1        10630          POP HL          ;EXECUTE ADDRESS
06CF 7D        10640          LD A,L
06D0 CDDD06    10650          CALL TOUT
06D3 7C        10660          LD A,H
06D4 CDDD06    10670          CALL TOUT
06D7 CD7901    10680          CALL PNHL       ;SHOW ADDRESS
06DA C31503    10690          JP TOFF1        ;TURN TAPE OFF
06DD D9        10700 TOUT     EXX             ;SAVE REGISTERS
06DE 0E08      10710          LD C,8
06E0 57        10720          LD D,A
06E1 37        10730 TOUT1    SCF             ;CARRY WILL OUTPUT A PULSE
06E2 CDF706    10740          CALL PULSE      ;OUT SYNC PULSE
06E5 7A        10750          LD A,D          ;GET DATA BIT
06E6 07        10760          RLCA            ;PUT IN CARRY
06E7 57        10770          LD D,A          ;RETURN TO D REGISTER
06E8 CDF706    10780          CALL PULSE      ;OUT WHATEVER
06EB 0D        10790          DEC C
06EC 20F3      10800          JR NZ,TOUT1
06EE 3A8038    10810          LD A,(3880H)    ;SHIFT PRESSED
06F1 B7        10820          OR A            ;SET FLAG
06F2 D9        10830          EXX             ;RESTORE REGISTERS
```

```
06F3 C21503   10840          JP NZ,TOFF1      ;IF NON-ZERO GET OUT
06F6 C9       10850          RET
06F7 3E00     10860 PULSE    LD A,00H         ;CLEAR AC WITHOUT CLEARING CARRY
06F9 17       10870          RLA              ;00 OR 01 DEPENDING UPON DATA
06FA D3FF     10880          OUT (0FFH),A
06FC 0610     10890          LD B,16D         ;SHORT DELAY
06FE 10FE     10900 P1       DJNZ P1
0700 E601     10910          AND 01H          ;KEEP INFORMATION BUT NO
0702 17       10920          RLA              ;CARRY
0703 D3FF     10930          OUT (0FFH),A     ;IF 1 ANOTHER PULSE
0705 0610     10940          LD B,16D         ;DELAY
0707 10FE     10950 P2       DJNZ P2
0709 AF       10960          XOR A            ;RETURN TO ZERO LEVEL
070A D3FF     10970          OUT (0FFH),A
070C 0674     10980          LD B,116D        ;REMAINDER OF MILLISECOND
070E 10FE     10990 P3       DJNZ P3
0710 C9       11000          RET
              11010 ;
              11020 ;        READ CASSETTE TAPE
              11030 ;
0711 CDE302   11040 READ     CALL GET
0714 28FB     11050          JR Z,READ        ;SKIP SPACES
0716 3807     11060          JR C,RD11
0718 FD2B     11070          DEC IY           ;ADJUST
071A CD3202   11080          CALL GHL
071D 1803     11090          JR RD22
071F 210000   11100 RD11     LD HL,00H
0722 E5       11110 RD22     PUSH HL          ;WILL POP TO DE
0723 3E02     11120          LD A,02H
0725 D3EC     11130          OUT (0ECH),A     ;TURN ON TAPE
0727 FD2B     11140          DEC IY           ;ADJUST
0729 CD9B07   11150 RD1      CALL BIT
072C FEA5     11160          CP 0A5H          ;SYNC?
072E 20F9     11170          JR NZ,RD1
0730 2A4040   11180          LD HL,(CURSOR)   ;GET CURSOR WANT LOAD FAST
0733 0606     11190          LD B,6H
0735 CD8F07   11200          CALL BYTE
0738 FE55     11210          CP 55H
073A 2038     11220          JR NZ,TOFF
073C CD8F07   11230 RD2      CALL BYTE
073F 77       11240          LD (HL),A
0740 23       11250          INC HL           ;ADJUST POINTER AND PROVIDE SPACE
0741 10F9     11260          DJNZ RD2
0743 224040   11270          LD (CURSOR),HL
0746 CD8F07   11280 RD3      CALL BYTE
0749 FE78     11290          CP 78H
074B 282A     11300          JR Z,END1
074D FE3C     11310          CP 03CH
074F 2023     11320          JR NZ,TOFF
0751 CD8F07   11330          CALL BYTE
0754 47       11340          LD B,A
0755 CD8F07   11350          CALL BYTE
0758 6F       11360          LD L,A
0759 CD8F07   11370          CALL BYTE
075C 67       11380          LD H,A
075D 22FE3F   11390          LD (3FFEH),HL
0760 85       11400          ADD A,L          ;START OF CHECKSUM
```

```
0761 4F       11410          LD C,A
0762 D1       11420          POP DE          ;GET BIAS FOR LOAD
0763 D5       11430          PUSH DE
0764 19       11440          ADD HL,DE       ;BIAS ADDED TO HL
0765 CD8F07   11450 RD4      CALL BYTE
0768 77       11460          LD (HL),A
0769 23       11470          INC HL
076A 81       11480          ADD A,C
076B 4F       11490          LD C,A
076C 10F7     11500          DJNZ RD4
076E CD8F07   11510          CALL BYTE       ;GET CHECKSUM
0771 B9       11520          CP C
0772 28D2     11530          JR Z,RD3
0774 C30003   11540 TOFF     JP ERROR
0777 CD8F07   11550 END1     CALL BYTE
077A 6F       11560          LD L,A
077B CD8F07   11570          CALL BYTE
077E 67       11580          LD H,A          ;HL HAS START ADDRESS
077F D1       11590          POP DE
0780 110000   11600          LD DE,00H       ;CORRECT STACK AND ZERO
0783 19       11610          ADD HL,DE
0784 CD7901   11620          CALL PNHL
0787 223C40   11630          LD (PCSAVE),HL  ;STORE TO PC PLACE
078A FD23     11640          INC IY          ;ADJUST BUFFER
078C C31503   11650          JP TOFF1        ;TURN TAPE OFF
078F C5       11660 BYTE     PUSH BC
0790 E5       11670          PUSH HL
0791 0608     11680          LD B,8H
0793 CD9B07   11690 BYTE1    CALL BIT
0796 10FB     11700          DJNZ BYTE1
0798 E1       11710          POP HL
0799 C1       11720          POP BC
079A C9       11730          RET
079B C5       11740 BIT      PUSH BC
079C F5       11750          PUSH AF
079D AF       11760          XOR A
079E D3FF     11770          OUT (0FFH),A    ;RESET LATCH BEFORE READ
07A0 3A8038   11780 BIT1     LD A,(3880H)
07A3 B7       11790          OR A            ;SET FLAG IF EITHER SHIFT PRESSED
07A4 20CE     11800          JR NZ,TOFF
07A6 DBFF     11810 BIT5     IN A,(0FFH)
07A8 17       11820          RLA
07A9 30F5     11830          JR NC,BIT1      ;WAIT FOR PULSE
07AB 066E     11840          LD B,6EH        ;SHORT DELAY
07AD 10FE     11850 BIT2     DJNZ BIT2
07AF AF       11860          XOR A
07B0 D3FF     11870          OUT (0FFH),A    ;RESET LATCH
07B2 0698     11880          LD B,98H        ;WINDOW TO LOOK FOR PULSE
07B4 10FE     11890 BIT3     DJNZ BIT3
07B6 DBFF     11900          IN A,(0FFH)     ;IN DATA A 1 IS IN POSITION (7)
07B8 4F       11910          LD C,A
07B9 F1       11920          POP AF
07BA CB11     11930          RL C
07BC 17       11940          RLA             ;ONE BIT ROTATE THROUGH AC
07BD C1       11950          POP BC
07BE C9       11960          RET
              11970          ;
```

```
                11980          ;FIND ONE OR TWO BYTES
                11990          ;
07BF 3E02       12000 FINDTO   LD A,02H         ;FLAG FOR FINDING TWO BYTES
07C1 1802       12010          JR FIND
07C3 3E01       12020 FINDON   LD A,01H         ;FLAG FOR FINDING ONE BYTE
07C5 324340     12030 FIND     LD (COUNT),A     ;SAVE FOR REFERENCE
07C8 CDFA01     12040          CALL G3N         ;GET ARGUMENTS
07CB 7B         12050 FND1     LD A,E           ;LOW ORDER BYTE OR ONLY BYTE
07CC EDA1       12060 FND2     CPI              ;IS IT THERE
07CE E0         12070          RET PO           ;FINISHED  AND NOT FOUND
07CF 20FB       12080          JR NZ,FND2       ;NOT THERE BUT CONTINUE SEARCH
07D1 3A4340     12090          LD A,(COUNT)     ;MORE THAN ONE
07D4 3D         12100          DEC A            ;IF ONE NOW ITS ZERO
07D5 2806       12110          JR Z,SHOW
07D7 7A         12120          LD A,D           ;NEED SECOND BYTE
07D8 BE         12130          CP (HL)          ;IT MUST BE NEXT. THE CPI INC HL
07D9 2802       12140          JR Z,SHOW        ;IT WAS THERE SO SHOW IT
07DB 18EE       12150          JR FND1          ;CONTINUE SEARCH START WITH FIRST
07DD 3A4340     12160 SHOW     LD A,(COUNT)     ;GET NUMBER, DESTROYED BY DEC
07E0 C5         12170          PUSH BC          ;SAVE
07E1 4F         12180          LD C,A           ;PREPARE FOR DISPLAY
07E2 0600       12190          LD B,00H
07E4 2B         12200          DEC HL           ;ADJUST HL
07E5 CD9503     12210          CALL DISP1       ;SHOW ADDRESS FOLLOWED BY DATA
07E8 C1         12220          POP BC           ;GET BYTES REMAINING
07E9 18E0       12230          JR FND1          ;CONTINUE GET ALL OCCURRENCES
0000            12240          END
00000 TOTAL ERRORS
```

```
ADDR   0180 02530   05500 05850 07700 07880 08340
ADDR1  0183 02540   07150 08380
ADDR2  0186 02550   07300
ADJUST 0175 02440   02390
ALL1   05AE 08880   08720 08750 08940
ALL3   0590 08730   08870
ASCII  0351 05480   05160
ASCII1 0354 05490   05570
ASCII2 035C 05520   05580
ASCII4 035D 05530
BACKW  052C 08200   07990
BIT    079B 11740   11150 11690
BIT1   07A0 11780   11830
BIT2   07AD 11850   11850
BIT3   07B4 11890   11890
BIT5   07A6 11810
BKSP   01D9 03130   03010
BOTMS  4026 00220   06730
BPNO   4045 00160   01730 01920 06530
BPOINT 4048 00170   01630 06660
BREAK  03F1 06390   06340
BREAK1 0401 06460   06590 06650
BREAKS 4080 00250   01810 06280
BUF    0127 01970   00930
BUF0   012C 01990
BUF1   012F 02010   02120 02170 02240
BUFFER 4080 00090   00250 00910 05060 05670 05750
```

```
BUFFR   0368 05620   05170
BYTE    078F 11660   11200 11230 11280 11330 11350 11370 11450
                     11510 11550 11570
BYTE1   0793 11690   11700
CAPIT   02BC 04570   04550
CD      008A 01050   01060 05790
CHAR    02CB 04660   04640
CHKIN   026D 04050   02850 03960
CLEAR   01F2 03270   03110 03310
CLEARS  01ED 03250   00790 05180
CLRBP   0107 01720   00890 01660 05080
CLRBP1  011D 01870   01900
CMARK   4044 00150   00840 03150
CMND    0080 01010   05000
COM1    006F 00910   06160
COMAND  006C 00900   04880
COMTAB  031B 05150   01090
CON     002B 00580   00550
CON1    0033 00620   00590
CON2    004C 00760   00630
CONTCP  0571 08610   08640
CONTI1  02F6 04970   04990
CONTIN  02F8 04980   01020 05110
COUNT   4043 00140   12030 12090 12160
CRT     01B2 02940   02160 02220 02650 05530 07940 08020 09070
                     09170 09360
CURSOR  4040 00120   02960 03140 11180 11270
D1      0279 04120   04120
DELETE  013E 02080
DISP1   0395 05850   01710 05910 12210
DISP2   0398 05860   05920
DISPLY  0392 05840   05190
DLT     0142 02100
DLTE    0147 02130   02200
DLTE1   0148 02140
END1    0777 11550   11300
ENT     0520 08120   08230
ENT1    0523 08150   08110
ENTER   03B4 06100   05220 05350
ER0     0305 05030   05050
ER1     030C 05060   05040
ER2     0317 05100
ERROR   0300 05010   01080 01120 03500 03700 05250 05260 05400
                     08630 11540
EXAM    0553 08470   05390 08480
EXAM2   0566 08560   08530
EXAM3   056A 08580
EXX     00CF 01450   01390
FILL    03A5 05980   05150
FILL1   03A9 06000   00780 06030
FINB    0688 10230   10160 10180 10250
FIND    07C5 12030   12010
FINDON  07C3 12020   05210
FINDTO  07BF 12000   05320
FINISH  06C1 10560   10290
FIRST   0530 08230   07970
FLAG1   05F6 09300   09380
```

```
FLAG2   0600 09360   09340
FLAGP   05EF 09270   08780 09630
FND1    07CB 12050   12150 12230
FND2    07CC 12060   12080
FOUND   0606 09390   08620
FOUND1  060C 09430   09450
FTAB    0631 09650   09290
G2N     0203 03400   03350 05480 05840 07250
G2N0    020C 03440   03450
G2N1    0222 03530   03470
G2N2    0229 03570   03520 03620
G2N3    022D 03610   03430
G3N     01FA 03350   05980 07490 08290 10010 12040
GBYTE   025D 03950   02010 03990 04020 07950
GET     02E3 04850   01050 03440 03670 03720 05030 05620 06300
                     06390 08470 08510 09430 10170 11040
GET1    02E5 04860   03420 03490 04980 06140 06290 07730 07810
                     10150
GET3    02F3 04930
GHL     0232 03660   03370 03400 03510 03530 06360 06450 06550
                     06600 07100 07120 07610 07630 07690 07720
                     07870 09470 11080
GHL0    0235 03670   03680
GHL1    0240 03710   03740
GNAM3   065F 10050
GO      03C7 06220   05200
GO1     03D9 06290   06380 06540
GO2     03DC 06300   06310
GO4     0447 06800   06780
HDATA   0295 04310   04250 04340
HEXM    046D 07100   05230
HLSAVE  403A 00240   01270 01330 06910
IN      0482 07250   05290
INCK    051D 08100   08010
INNEXT  0487 07280   07390
INS1    04EA 07880   08170 08190 08220
INS2    04ED 07890
INS3    04FB 07950   08090
INSERT  04E7 07870   05340
JMP     42A0 00350   06680 06700 06720 07040 07060
KYD1    0283 04210   04090 04130
LDHL    0197 02710   03710 08080
LDREG   0424 06660   06320
LEAD    0664 10080   10090
LINEF   01C7 03060   02990
LIST    04A1 07410   05270
LOWBIT  029B 04350   04320
MINUS   0418 06600   06430
MONSP   42C0 00340   00350 00880 01010 01670
MOV1    04AF 07500
MOVE    04AC 07490   05280
NADDS   0529 08180   08040
NALPH1  02C3 04610   04700 04740
NALPHA  02BF 04590   04500
NEWAD1  04E2 07840   07750
NEWADR  04E4 07850
NEWLIN  019E 02810   01040 02530 08740
```

```
NEXT    0289 04230    04280
NOSFT   02AD 04470    04450
NOT7    02CE 04680    04600
NOTTWO  0622 09560    09440 09520
NPRIM   05DA 09160    09100
OFLAG   05FF 09350    09310
OLD     4047 00270    03970
ONEB    05EA 09240    09190
OUT     04BA 07610    05300
OUT2    01DA 03140    03050
P1      06FE 10900    10900
P2      0707 10950    10950
P3      070E 10990    10990
PCH     0150 02180    02070
PCHAR   0152 02190    02090
PCP     0657 09960    08670
PCSAVE  403C 00230    00240 00290 01250 01420 01440 01690 06370
                      06570 06620 06710 11630
PHEX    0160 02310    02500 02520 07290 07330 09220 09250
PHEX1   0170 02410    02450
PHEXH   0163 02320    02310
PHEXL   0167 02360
PLUS    040E 06550    06410
PMSG    018D 02620    02670 05070
PMSG1   018A 02610    00900
PNHL    0179 02490    02540 07200 10680 11620
POPAF   0469 07050    06950
PPHL    015F 02300
PRIME   05DC 09170    09150
PRIMF   404B 00190    08570 08660 08810 08830 09080
PRINIT  04F6 07930    08060
PRNT    01A4 02840    02410 02560 02820
PRNT1   01A5 02850    02890
PRNTFG  404A 00180    03190 07420 07440
PROM    401B 00310    05310
PULSE   06F7 10860    10740 10780
PVAL    05BB 08960    08680 08890
PVAL1   05BE 08990    09420 09580
RA      0645 09780    08860
RD1     0729 11150    11170
RD11    071F 11100    11060
RD2     073C 11230    11260
RD22    0722 11110    11090
RD3     0746 11280    11530
RD4     0765 11450    11500
READ    0711 11040    05330 11050
REF     4000 00080    00090 00100 00110 00200 00270 00280 00300
                      00310 00320 00330 00410
REF1    4040 00110    00120 00130 00140 00150 00160 00170 00180
                      00190
REF2    4024 00200    00210 00230
RSAVE   00A6 01230    06260
RST     00FF 00070    06490
RSTART  0066 00880    02030 02870
RSTLOC  4012 00060    06240 06270
S1      000B 00440    00420
S2      0013 00470    00450 00560
```

```
S3      001B 00510   00490
S4      0023 00550   00530
SAVE    0373 05670   05650
SAVE1   00DE 01560   01540
SCREEN  403E 00290   00810 01610 06100
SHOW    07DD 12160   12110 12140
SIGNON  0043 00730   02610
SPACE   01A2 02830   02290 05510 07310 07340 07350 08370 08690
                     08910 09280 09560
SPCHL   015C 02290   05860 07790 07890 08150 08350 08390
SPEC    02DB 04750   04360
SSFLAG  4042 00130   01370 01410 06080 06930
SSTACK  4024 00210   00220 00820 01310 06890 08990
STEP    03AF 06070   05410
SUBS    04C7 07700   07860
SUBS1   04CA 07710   07830
SUBS3   04D5 07760
SUBSM   04C4 07690   05240
TABLE   003B 00650   04750
TDATA   069B 10330   10310 10600
TDATA1  06B4 10470   10530
TEM     4180 00100   05690 05730
TOFF    0774 11540   11220 11320 11800
TOFF1   0315 05090   10690 10840 11650
TOUT    06DD 10700   10080 10110 10130 10190 10240 10350 10370
                     10390 10410 10480 10550 10620 10650 10670
TOUT1   06E1 10730   10800
TRAN    0381 05730   05630
TTYP    4018 00300   00770 03210
TWO     0273 04090   04150
UCFLAG  4046 00280   01980 04510
USER    401E 00320   05360
USERSP  4300 00330   00340 00620
VALHEX  024C 03800   03690 03730 08050
VERF2   054F 08420   08320
VERIF1  0537 08300   07570 08440
VERIFY  0534 08290   05370
VIDEO   3C00 00260   03070 03080 03120 03250
WNAME   067A 10170   10200
WRIT    0659 10010   05380
WRIT1   0669 10100
WRIT2   0691 10280   10320
WRIT3   068F 10260   10220
WRIT5   06C9 10610   10580
XALL    057B 08650   01680 08490
XTAB    0639 09660   08590 08700
```

APPENDIX F

Source Listing for the 2708 EPROM Programmer

```
                00010 ;       PROM - BURN A 2708
                00020 ;       DISK NAME    P2708/ASM
                00030 ;       FOR MODIFIED TRS-80 SYSTEMS
                00040 ;
                00050 ;       COMMANDS:
                00060 ;       READ:     R  MEMORY ADDR., # BYTES, PROM ADDR.
                00070 ;       VERIFY:   V       "       ,     "   ,      "
                00080 ;       PROGRAM:  P       "       ,     "   ,      "
                00090 ;       SHORTP:   S       "       ,     "   ,      "
                00100 ;       EXIT:     E
                00110 ;
                00120 ;       SUBROUTINES USED FROM MONITOR:
                00130 ;       BUF, G3N, PMSG
                00140 ;
443F            00150         ORG 0443FH
                00160 ;
443F FDE5       00170         PUSH IY           ;SAVE MONITOR POINTER
4441 CD8946     00180         CALL SR           ;SET UP AND REMOVE VOLTAGES
4444 AF         00190 PR10    XOR A             ;CLEAR SHORT PROGRAM CYCLE FLAG
4445 32B547     00200         LD (SPFLAG),A
4448 21B846     00210         LD HL,MES1        :PRINT 'PROM PROGRAMMING PROGRAM'
444B CD8D01     00220         CALL PMSG
444E 21D946     00230         LD HL,MES2        :PRINT PROMPT (>)
4451 CD8D01     00240         CALL PMSG
4454 21BE47     00250         LD HL,BUFFER
4457 E5         00260         PUSH HL
4458 CD2701     00270         CALL BUF          :GET DATA
445B 3620       00280         LD (HL),20H       :PUT SPACE
445D 23         00290         INC HL
445E 3630       00300         LD (HL),30H       ;IN CASE THIRD ARGUMENT NOT ENTERED
4460 23         00310         INC HL
4461 362C       00320         LD (HL),','       ;END
4463 FDE1       00330         POP IY
                00340 ;
                00350 ;       BRANCH TO CORRECT SECTION OF PROGRAM DEPENDING
                00360 ;       ON COMMAND ENTERED
                00370 ;
4465 FD23       00380 PR20    INC IY
4467 FD7E00     00390         LD A,(IY+0)
446A FE52       00400         CP 'R'
446C CA9C44     00410         JP Z,RCMD
446F FE56       00420         CP 'V'
4471 CAA544     00430         JP Z,VCMD
4474 FE50       00440         CP 'P'
4476 CAAE44     00450         JP Z,PCMD
4479 FE53       00460         CP 'S'
447B 2812       00470         JR Z,SPROG
447D FE45       00480         CP 'E'
447F 2816       00490         JR Z,EXIT         ;IF EXIT, RETURN TO MONITOR
4481 FE2C       00500         CP ','            ;IF COMMA, PRINT 'UNKNOWN COMMAND'
4483 C26544     00510         JP NZ,PR20
4486 216547     00520         LD HL,MESA
4489 CD8D01     00530         CALL PMSG
448C C34444     00540         JP PR10
448F 3E01       00550 SPROG   LD A,01H          ;SET FLAG
4491 32B547     00560         LD (SPFLAG),A
4494 C3AE44     00570         JP PCMD
```

```
4497 FDE1      00580 EXIT    POP IY
4499 FD23      00590         INC IY
449B C9        00600         RET              ;BUFFER POINTER ADJUSTED
               00610 ;
               00620 ;
               00630 ;       R - READ
               00640 ;
449C CD0B02    00650 RCMD    CALL G3N         ;GET DATA *ADDRESS CHANGE MODEL III*
449F CD7745    00660         CALL READ        ;SUBROUTINE READ DOES THE WORK
44A2 C34444    00670         JP PR10
               00680 :
               00690 ;       V - VERIFY
               00700 ;
44A5 CD0B02    00710 VCMD    CALL G3N         ;GET DATA *ADDRESS CHANGE MODEL III*
44A8 CD9045    00720         CALL VERIFY      ;SUBROUTINE VERIFY DOES THE WORK
44AB C34444    00730         JP PR10
               00740 :
               00750 ;       P - PROGRAM
               00760 ;
44AE CD0B02    00770 PCMD    CALL G3N         ;*ADDRESS CHANGE MODEL III*
44B1 ED43B847  00780         LD (NBYTES),BC   ;SAVE ARGUMENTS
44B5 ED53BA47  00790         LD (PROMA),DE
44B9 22BC47    00800         LD (MEMA),HL
44BC 010004    00810         LD BC,1024       ;READ THE WHOLE
44BF 210148    00820         LD HL,FWM        ;PROM INTO MEMORY
44C2 110000    00830         LD DE,00H
44C5 CD7745    00840         CALL READ
               00850 :
               00860 ;       CHECK IF IT'S DIRTY
               00870 ;
44C8 210148    00880         LD HL,FWM
44CB 010004    00890         LD BC,1024
44CE 7E        00900 P10     LD A,(HL)
44CF FEFF      00910         CP 0FFH
44D1 C2E444    00920         JP NZ,P20        ;IT'S DIRTY
44D4 23        00930         INC HL
44D5 0B        00940         DEC BC
44D6 78        00950         LD A,B
44D7 B1        00960         OR C
44D8 C2CE44    00970         JP NZ,P10
44DB 217647    00980         LD HL,MESB
44DE CD8D01    00990         CALL PMSG
44E1 C34045    01000         JP P80           ;IT'S CLEAN
44E4 211247    01010 P20     LD HL,MES5
44E7 CD8D01    01020         CALL PMSG        ;PRINT 'DIRTY PROM.'
               01030 :
               01040 ;       DETERMINE WHETHER IT CAN BE PROGRAMMED
               01050 ;       IF ATTEMPT IS MADE TO CHANGE A 0 TO 1,
               01060 ;       IT WON'T WORK
               01070 ;
44EA 210148    01080         LD HL,FWM
44ED ED5BBA47  01090         LD DE,(PROMA)
44F1 19        01100         ADD HL,DE        ;DE = ADDRESS OF PROM DATA
44F2 EB        01110         EX DE,HL
44F3 2ABC47    01120         LD HL,(MEMA)
44F6 ED4BB847  01130         LD BC,(NBYTES)
44FA 1A        01140 P30     LD A,(DE)
```

```
44FB 2F        01150          CPL
44FC A6        01160          AND (HL)
44FD C23745    01170          JP NZ,PNO        ;IF NZ IT WON'T WORK
4500 13        01180          INC DE
4501 23        01190          INC HL
4502 0B        01200          DEC BC
4503 78        01210          LD A,B
4504 B1        01220          OR C
4505 C2FA44    01230          JP NZ,P30
4508 213547    01240          LD HL,MES7       ;PRINT 'IT COULD WORK'
450B CD8D01    01250 P50      CALL PMSG
450E 214647    01260          LD HL,MES8       ;PRINT 'CONTINUE?'
4511 CD8D01    01270          CALL PMSG
4514 21BE47    01280 P60      LD HL,BUFFER
4517 CD2701    01290          CALL BUF
451A 21BE47    01300          LD HL,BUFFER
451D 23        01310 P70      INC HL           ;CHECK IF YES OR NO TYPED IN
451E 7E        01320          LD A,(HL)
451F FE59      01330          CP 'Y'           ;IF YES, CONTINUE
4521 CA4045    01340          JP Z,P80
4524 FE4E      01350          CP 'N'
4526 CA4444    01360          JP Z,PR10        ;IF NO, GO BACK TO BEGINNING
4529 FE0D      01370          CP 0DH
452B C21D45    01380          JP NZ,P70
452E 215147    01390          LD HL,MES9       ;PRINT 'ANSWER YES OR NO'
4531 CD8D01    01400          CALL PMSG
4534 C31445    01410          JP P60
4537 212147    01420 PNO      LD HL,MES6       ;PRINT 'IT WON'T WORK'
453A CD8D01    01430          CALL PMSG
453D C34444    01440          JP PR10          ;GO BACK FOR NEW COMMAND
               01450 ;
               01460 ;        ATTEMPT TO PROGRAM THE 2708
               01470 ;
4540 218247    01480 P80      LD HL,MESC       ;PRINT'WAIT FOR PROGRAMMING
               01490                           ; MAX TIME 2 MINUTES'
4543 CD8D01    01500          CALL PMSG
4546 210148    01510          LD HL,FWM
4549 ED5BBA47  01520          LD DE,(PROMA)    ;DE = PROMA + FWM
454D 19        01530          ADD HL,DE
454E EB        01540          EX DE,HL
454F 2ABC47    01550          LD HL,(MEMA)
4552 ED4BB847  01560          LD BC,(NBYTES)
4556 EDB0      01570          LDIR             ;COPY MEMORY INTO BUFFER AREA
4558 3AB547    01580          LD A,(SPFLAG)    ;SHORT CYCLE REQUEST
455B B7        01590          OR A             ;SET FLAG
455C 2014      01600          JR NZ,SHORT
455E CD0D46    01610          CALL PROG        ;PROGRAM IT
4561 2ABC47    01620 P90      LD HL,(MEMA)
4564 ED5BBA47  01630          LD DE,(PROMA)
4568 ED4BB847  01640          LD BC,(NBYTES)
456C CD9045    01650          CALL VERIFY
456F C34444    01660          JP PR10
4572 CD4A46    01670 SHORT    CALL PGS         ;REPEAT PROGRAM CYCLE TO VERIFY
4575 18EA      01680          JR P90
               01690 ;
               01700 ;        READ - READ PROM INTO MEMORY
               01710 ;
```

```
                01720 ;         BC = # BYTES TO READ
                01730 ;         DE = PROM ADDRESS
                01740 ;         HL = MEMORY ADDRESS
                01750 ;
4577 CD8946     01760 READ      CALL SR           ;SET UP CHIP TO READ
457A 7B         01770 RD10      LD A,E
457B D38D       01780           OUT (PB),A        ;OUTPUT A0-A7
457D 7A         01790           LD A,D
457E F670       01800           OR 70H
4580 D38E       01810           OUT (PC),A        ;OUTPUT A8-A10
4582 C5         01820           PUSH BC
4583 C1         01830           POP BC
4584 DB8C       01840           IN A,(PA)         ;GET DATA BYTES
4586 77         01850           LD (HL),A         ;STORE IT IN MEMORY
4587 13         01860           INC DE
4588 23         01870           INC HL            ;GET READY TO READ NEXT BYTE
4589 0B         01880           DEC BC
458A 78         01890           LD A,B
458B B1         01900           OR C
458C C27A45     01910           JP NZ,RD10
458F C9         01920           RET
                01930 ;
                01940 ;         VERIFY - VERIFY MEMORY AGAINST PROM MEMORY
                01950 ;
                01960 ;         HL = MEMORY ADDRESS
                01970 ;         DE = PROM MEMORY ADDRESS
                01980 ;         BC = # BYTES TO VERIFY
                01990 ;
4590 E5         02000 VERIFY    PUSH HL
4591 210000     02010           LD HL,0000H       ;ERRS = 0
4594 22B647     02020           LD (ERRS),HL
4597 E1         02030           POP HL
                02040 ;
4598 CD8946     02050           CALL SR           ;SET UP CHIP TO READ
459B 7B         02060 VE10      LD A,E
459C D38D       02070           OUT (PB),A        ;OUTPUT A0-A7
459E 7A         02080           LD A,D
459F F670       02090           OR 70H
45A1 D38E       02100           OUT (PC),A        ;OUTPUT A8-A10
45A3 C5         02110           PUSH BC
45A4 C1         02120           POP BC
45A5 DB8C       02130           IN A,(PA)
45A7 BE         02140           CP (HL)
45A8 CAE945     02150           JP Z,VE100        ;IF MATCH, INCREMENT POINTERS
                02160 ;
                02170 ;         THERE IS A DISCREPANCY
                02180 ;         WRITE 'MEMORY: XXXX XX   PROM: XXXX XX'
                02190 ;
45AB C5         02200           PUSH BC
45AC E5         02210           PUSH HL
45AD D5         02220           PUSH DE
45AE F5         02230           PUSH AF
45AF 11E546     02240           LD DE,MES31       ;FORMAT MEMORY ADDRESS
45B2 7C         02250           LD A,H
45B3 CD9446     02260           CALL I2AH
45B6 13         02270           INC DE
45B7 13         02280           INC DE
```

```
45B8 7D        02290          LD A,L
45B9 CD9446    02300          CALL I2AH
45BC 11EA46    02310          LD DE,MES32      ;FORMAT MEMORY CONTENT
45BF 7E        02320          LD A,(HL)
45C0 CD9446    02330          CALL I2AH
45C3 11FB46    02340          LD DE,MES34      ;FORMAT PROM CONTENTS
45C6 F1        02350          POP AF
45C7 CD9446    02360          CALL I2AH
45CA 11F646    02370          LD DE,MES33
45CD E1        02380          POP HL           ;GET PROM MEMORY ADDRESS
45CE E5        02390          PUSH HL
45CF 7C        02400          LD A,H
45D0 CD9446    02410          CALL I2AH        ;FORMAT PROM MEMORY ADDRESS
45D3 13        02420          INC DE
45D4 13        02430          INC DE
45D5 7D        02440          LD A,L
45D6 CD9446    02450          CALL I2AH
45D9 2AB647    02460          LD HL,(ERRS)     ;ERRS = ERRS + 1
45DC 23        02470          INC HL
45DD 22B647    02480          LD (ERRS),HL
45E0 21DC46    02490          LD HL,MES3
45E3 CD8D01    02500          CALL PMSG        ;PRINT THE DISCREPANCY
45E6 D1        02510          POP DE
45E7 E1        02520          POP HL
45E8 C1        02530          POP BC
               02540 ;
45E9 13        02550 VE100    INC DE
45EA 23        02560          INC HL
45EB 0B        02570          DEC BC
45EC 3A8038    02580          LD A,(3880H)
45EF B7        02590          OR A             ;SHIFT FOR ESCAPE
45F0 C0        02600          RET NZ
45F1 78        02610          LD A,B
45F2 B1        02620          OR C
45F3 C29B45    02630          JP NZ,VE10
               02640 ;
45F6 2AB647    02650          LD HL,(ERRS)     ;FORMAT ERRS
45F9 11FF46    02660          LD DE,MES41
45FC 7C        02670          LD A,H
45FD CD9446    02680          CALL I2AH
4600 13        02690          INC DE
4601 13        02700          INC DE
4602 7D        02710          LD A,L
4603 CD9446    02720          CALL I2AH
4606 21FE46    02730          LD HL,MES4       ;PRINT 'XXXX DISCREPANCIES'
4609 CD8D01    02740          CALL PMSG
460C C9        02750          RET
               02760 ;
               02770 ;        PROG - WRITE MEMORY INTO PROM
               02780 ;
               02790 ;        ALL 1024 WORDS OF THE DATA BUFFER ARE
               02800 ;        WRITTEN TO THE PROM.  TIME APPROXIMATELY 2 MINUTES
               02810 ;
460D 016400    02820 PROG     LD BC,100D       ;NUMBER OF PROGRAM LOOPS
4610 3E80      02830 PROGC    LD A,80H         ;MODE 0, ALL PORTS = OUTPUT
4612 D38F      02840          OUT (P0),A
4614 3E20      02850          LD A,20H
```

```
4616 D38E      02860           OUT (PC),A      ;SET WRITE ENABLE
4618 21004C    02870 PG10      LD HL,LWM       ;HL = ADDR OF LAST WORD IN BUFFER
461B 11FF03    02880           LD DE,1023
461E 7E        02890 PG20      LD A,(HL)
461F D38C      02900           OUT (PA),A      ;OUTPUT D0-D7
4621 7B        02910           LD A,E
4622 D38D      02920           OUT (PB),A      ;OUTPUT A0-A7
4624 7A        02930           LD A,D
4625 F620      02940           OR 20H
4627 D38E      02950           OUT (PC),A      ;OUTPUT A8-A10
4629 F5        02960           PUSH AF
462A F1        02970           POP AF          ;SHORT DELAY
462B EE20      02980           XOR 20H
462D D38E      02990           OUT (PC),A      ;PULSE IT
462F C5        03000           PUSH EC
               03010 ;
               03020 ;         CRITICAL TIME - MODEL DEPENDENT
               03030 ;
4630 0685      03040           LD B,133D       ;** MODEL III 152D OR 98H **
4632 10FE      03050 PG30      DJNZ PG30       ;WAIT 1.0 MILLISECONDS
4634 C1        03060           POP BC
4635 EE20      03070           XOR 20H
4637 D38E      03080           OUT (PC),A      ;END OF PULSE
4639 3A8038    03090           LD A,(3880H)
463C B7        03100           OR A            ;SHIFT FOR ESCAPE
463D C0        03110           RET NZ
463E 7A        03120           LD A,D
463F B3        03130           OR E
4640 2B        03140           DEC HL
4641 1B        03150           DEC DE          ;FINISHED A PROGRAM LOOP?
4642 C21E46    03160           JP NZ,PG20
4645 0D        03170           DEC C           ;FINISHED WITH LOOP?
4646 C21846    03180           JP NZ,PG10
4649 C9        03190           RET
               03200 ;
               03210 ;         SHORT -  SHORT PROGRAM CYCLE
               03220 ;
464A 010100    03230 PGS       LD BC,0001H     ;INTIALIZE COUNT
464D CD1046    03240 PGS10     CALL PROGC      ;ONE LOOP DEFINED BY C
4650 04        03250           INC B           ;KEEP TRACK OF LOOPS
4651 3E44      03260           LD A,68D        ;NO LONGER THAN LONG PROGRAM
4653 B8        03270           CP B
4654 CA6946    03280           JP Z,PGS20      ;AFTER 256, BAD PROM
4657 0C        03290           INC C           ;SET UP FOR 1 PASS
4658 C5        03300           PUSH BC         ;SAVE
4659 CD8946    03310           CALL SR         ;SET UP TO READ
465C 3A8038    03320           LD A,(3880H)
465F B7        03330           OR A            ;SHIFT FOR ESCAPE
4660 C0        03340           RET NZ
4661 13        03350           INC DE          ;CORRECT DE TO 0000H
4662 23        03360           INC HL          ;ADJUST HL ALSO
4663 CD6E46    03370           CALL CHECK
4666 C1        03380           POP BC
4667 20E4      03390           JR NZ,PGS10     ;NO VERIFICATION, TRY AGAIN
4669 48        03400 PGS20     LD C,B          ;REPEAT FOR HALF NUMBER
466A CB39      03410           SRL C
466C 18A2      03420           JR PROGC        ;PROGRAM REMAINING CYCLES
```

```
                03430 ;
                03440 ;       CHECK - LIKE VERIFY BUT NO MESSAGES
                03450 ;
466E 010004     03460 CHECK   LD BC,0400H     ;CHECK ALL
4671 7B         03470 CH1     LD A,E
4672 D38D       03480         OUT (PB),A      ;LOW ADDRESS
4674 7A         03490         LD  A,D
4675 F670       03500         OR 70H
4677 D38E       03510         OUT (PC),A      ;HIGH ADDRESS AND CONTROL
4679 C5         03520         PUSH BC
467A C1         03530         POP BC
467B C5         03540         PUSH BC
467C C1         03550         POP BC
467D DB8C       03560         IN A,(PA)       ;READ PROM
467F BE         03570         CP (HL)
4680 C0         03580         RET NZ          ;RETURN IF NO CHECK
4681 13         03590         INC DE
4682 23         03600         INC HL
4683 0B         03610         DEC BC
4684 78         03620         LD A,B
4685 B1         03630         OR C
4686 20E9       03640         JR NZ,CH1
4688 C9         03650         RET
                03660 ;
                03670 ;       SR - SET UP CHIP TO READ
                03680 ;
4689 3E90       03690 SR      LD A,90H
468B D38F       03700         OUT (P0),A      ;MODE 0, PORT A = INPUT
468D 3E30       03710         LD A,30H        ;PORT B + C = OUTPUT
468F D38E       03720         OUT (PC),A
4691 C5         03730         PUSH BC
4692 C1         03740         POP BC          ;SHORT DELAY
4693 C9         03750         RET
                03760 ;
                03770 ;       I2AH - INTEGER TO ASCII HEX
                03780 ;
                03790 ;       A = INTEGER TO CONVERT
                03800 ;       DE = ADDRESS OF ASCII BUFFER
                03810 ;       ONLY THE FLAG REGISTER IS ALTERED
                03820 ;
4694 C5         03830 I2AH    PUSH BC
4695 13         03840         INC DE          ;CONVERT LAST NIBBLE FIRST
4696 47         03850         LD B,A          ;SAVE THE INTEGER
4697 E60F       03860         AND 0FH
4699 FE0A       03870         CP 0AH
469B FAA046     03880         JP M,IH10
469E C607       03890         ADD A,07        ;ADD 7 FOR A-F
46A0 C630       03900 IH10    ADD A,30H
46A2 12         03910         LD (DE),A
46A3 1B         03920         DEC DE          ;NOW CONVERT FIRST NIBBLE
46A4 78         03930         LD A,B
46A5 07         03940         RLCA            ;PUT LEFT NIBBLE
46A6 07         03950         RLCA            ; IN RIGHT NIBBLE
46A7 07         03960         RLCA
46A8 07         03970         RLCA
46A9 E60F       03980         AND 0FH
46AB FE0A       03990         CP 0AH
```

```
46AD FAB246   04000          JP M,IH20
46B0 C607     04010          ADD A,7          ;ADD 7 FOR A-F
46B2 C630     04020 IH20     ADD A,30H
46B4 12       04030          LD (DE),A
46B5 78       04040          LD A,B           ;RESTORE REGISTERS
46B6 C1       04050          POP BC
46B7 C9       04060          RET
              04070 ;
46B8 0D       04080 MES1     DEFB 0DH
46B9 0D       04090          DEFB 0DH
46BA 32       04100          DEFM '2708 EPROM PROGRAMMING PROGRAM'
46D8 01       04110          DEFB 01H
46D9 0D       04120 MES2     DEFB 0DH
46DA 3E       04130          DEFM '>'
46DB 01       04140          DEFB 01H
46DC 0D       04150 MES3     DEFB 0DH
46DD 4D       04160          DEFM 'MEMORY: '
0004          04170 MES31    DEFS 4
46E9 20       04180          DEFM ' '
0002          04190 MES32    DEFS 2
46EC 20       04200          DEFM '    PROM: '
0004          04210 MES33    DEFS 4
46FA 20       04220          DEFM ' '
0002          04230 MES34    DEFS 2
46FD 01       04240          DEFB 01H
46FE 0D       04250 MES4     DEFB 0DH
0004          04260 MES41    DEFS 4
4703 20       04270          DEFM ' DISCREPANCIES'
4711 01       04280          DEFB 01H
4712 0D       04290 MES5     DEFB 0DH
4713 44       04300          DEFM 'DIRTY PROM.  '
4720 01       04310          DEFB 01H
4721 49       04320 MES6     DEFM 'IT WILL NOT WORK.  '
4734 01       04330          DEFB 01H
4735 49       04340 MES7     DEFM 'IT COULD WORK.  '
4745 01       04350          DEFB 01H
4746 43       04360 MES8     DEFM 'CONTINUE? '
4750 01       04370          DEFB 01H
4751 0D       04380 MES9     DEFB 0DH
4752 41       04390          DEFM 'ANSWER YES OR NO: '
4764 01       04400          DEFB 01H
4765 0D       04410 MESA     DEFB 0DH
4766 55       04420          DEFM 'UNKNOWN COMMAND'
4775 01       04430          DEFB 01H
4776 0D       04440 MESB     DEFB 0DH
4777 43       04450          DEFM 'CLEAN PROM'
4781 01       04460          DEFB 01H
4782 0D       04470 MESC     DEFB 0DH
4783 57       04480          DEFM 'WAIT FOR PROGRAMMING'
4797 0D       04490          DEFB 0DH
4798 4D       04500          DEFM 'MAXIMUM TIME ABOUT 2 MINUTES'
47B4 01       04510          DEFB 01H
              04520 ;
0001          04530 SPFLAG   DEFS 1
0002          04540 ERRS     DEFS 2
0002          04550 NBYTES   DEFS 2
0002          04560 PROMA    DEFS 2
```

```
0002            04570 MEMA      DEFS 2
0043            04580 BUFFER    DEFS 67
0400            04590 FWM       DEFS 1024
4C00            04600 LWM       EQU FWM+1023
                04610 ;
                04620 ;         MONITOR SUBROUTINES
                04630 ;
0127            04640 BUF       EQU 0127H       ;MODEL III NO CHANGE
018D            04650 PMSG      EQU 018DH       ;MODEL III NO CHANGE
020B            04660 G3N       EQU 020BH       ;MODEL III EQU 01FAH
008C            04670 PA        EQU 8CH
008D            04680 PB        EQU 8DH
008E            04690 PC        EQU 8EH
008F            04700 P0        EQU 8FH
0000            04710           END
00000 TOTAL ERRORS


BUF     0127 04640    00270 01290
BUFFER 47BE 04580     00250 01280 01300
CH1     4671 03470    03640
CHECK   466E 03460    03370
ERRS    47B6 04540    02020 02460 02480 02650
EXIT    4497 00580    00490
FWM     4801 04590    00820 00880 01080 01510 04600
G3N     020B 04660    00650 00710 00770
I2AH    4694 03830    02260 02300 02330 02360 02410 02450 02680
                      02720
IH10    46A0 03900    03880
IH20    46B2 04020    04000
LWM     4C00 04600    02870
MEMA    47BC 04570    00800 01120 01550 01620
MES1    46B8 04080    00210
MES2    46D9 04120    00230
MES3    46DC 04150    02490
MES31   46E5 04170    02240
MES32   46EA 04190    02310
MES33   46F6 04210    02370
MES34   46FB 04230    02340
MES4    46FE 04250    02730
MES41   46FF 04260    02660
MES5    4712 04290    01010
MES6    4721 04320    01420
MES7    4735 04340    01240
MES8    4746 04360    01260
MES9    4751 04380    01390
MESA    4765 04410    00520
MESB    4776 04440    00980
MESC    4782 04470    01480
NBYTES 47B8 04550     00780 01130 01560 01640
P0      008F 04700    02840 03700
P10     44CE 00900    00970
P20     44E4 01010    00920
P30     44FA 01140    01230
P50     450B 01250
P60     4514 01280    01410
P70     451D 01310    01380
```

```
P80     4540 01480   01000 01340
P90     4561 01620   01680
PA      008C 04670   01840 02130 02900 03560
PB      008D 04680   01780 02070 02920 03480
PC      008E 04690   01810 02100 02860 02950 02990 03080 03510
                     03720
PCMD    44AE 00770   00450 00570
PG10    4618 02870   03180
PG20    461E 02890   03160
PG30    4632 03050   03050
PGS     464A 03230   01670
PGS10   464D 03240   03390
PGS20   4669 03400   03280
PMSG    018D 04650   00220 00240 00530 00990 01020 01250 01270
                     01400 01430 01500 02500 02740
PNO     4537 01420   01170
PR10    4444 00190   00540 00670 00730 01360 01440 01660
PR20    4465 00380   00510
PROG    460D 02820   01610
PROGC   4610 02830   03240 03420
PROMA   47BA 04560   00790 01090 01520 01630
RCMD    449C 00650   00410
RD10    457A 01770   01910
READ    4577 01760   00660 00840
SHORT   4572 01670   01600
SPFLAG  47B5 04530   00200 00560 01580
SPROG   448F 00550   00470
SR      4689 03690   00180 01760 02050 03310
VCMD    44A5 00710   00430
VE10    459B 02060   02630
VE100   45E9 02550   02150
VERIFY  4590 02000   00720 01650
```

APPENDIX G

Source Listing for the 2716 EPROM Programmer

```
                00010 ;       PROM - BURN A 2716
                00020 ;       DISK NAME   P2716/ASM
                00030 ;       FOR MODIFIED TRS-80 SYSTEMS
                00040 ;
                00050 ;       COMMANDS:
                00060 ;       READ:     R  MEMORY ADDR., # BYTES, PROM ADDR.
                00070 ;       VERIFY:   V       "       ,     "   ,      "
                00080 ;       PROGRAM:  P       "       ,     "   ,      "
                00090 ;       EXIT:     E
                00100 ;
                00110 ;       SUBROUTINES USED FROM MONITOR:
                00120 ;       BUF, G3N, PMSG
                00130 ;
4400            00140         ORG 04400H
                00150 ;
4400 FDE5       00160         PUSH IY           ;SAVE MONITOR POINTER
4402 CDF845     00170         CALL SR           ;SET UP AND REMOVE VOLTAGES
4405 212746     00180 PR10    LD HL,MES1        ;PRINT 'PROM PROGRAMMING PROGRAM'
4408 CD8D01     00190         CALL PMSG
440B 214846     00200         LD HL,MES2        ;PRINT PROMPT (>)
440E CD8D01     00210         CALL PMSG
4411 214547     00220         LD HL,BUFFER
4414 E5         00230         PUSH HL
4415 CD2701     00240         CALL BUF
4418 3620       00250         LD (HL),20H       ;PUT SPACE
441A 23         00260         INC HL
441B 3630       00270         LD (HL),30H       ;IN CASE THIRD ARGUMENT NOT ENT
441D 23         00280         INC HL
441E 362C       00290         LD (HL),','       ;END
4420 FDE1       00300         POP IY
                00310 ;
                00320 ;       BRANCH TO CORRECT SECTION OF PROGRAM DEPENDING
                00330 ;       ON COMMAND ENTERED
                00340 ;
4422 FD23       00350 PR20    INC IY
4424 FD7E00     00360         LD A,(IY+0)
4427 FE52       00370         CP 'R'
4429 CA4D44     00380         JP Z,RCMD
442C FE56       00390         CP 'V'
442E CA5644     00400         JP Z,VCMD
4431 FE50       00410         CP 'P'
4433 CA5F44     00420         JP Z,PCMD
4436 FE45       00430         CP 'E'
4438 280E       00440         JR Z,EXIT         ;IF EXIT, RETURN TO MONITOR
443A FE2C       00450         CP ','            ;IF COMMA, PRINT 'UNKNOWN COMMAND'
443C C22244     00460         JP NZ,PR20
443F 21D446     00470         LD HL,MESA
4442 CD8D01     00480         CALL PMSG
4445 C30544     00490         JP PR10
4448 FDE1       00500 EXIT    POP IY
444A FD23       00510         INC IY            ;ADJUST BUFFER
444C C9         00520         RET
                00530 ;
                00540 ;       R - READ
                00550 ;
444D CD0B02     00560 RCMD    CALL G3N          ;*ADDRESS CHANGE MODEL III*
4450 CD1E45     00570         CALL READ         ;SUBROUTINE READ DOES THE WORK
```

```
4453 C30544     00580           JP PR10
                00590 ;
                00600 ;         V - VERIFY
                00610 ;
4456 CD0B02     00620 VCMD      CALL G3N          ;*ADDRESS CHANGE MODEL III*
4459 CD3745     00630           CALL VERIFY       ;SUBROUTINE VERIFY DOES THE WORK
445C C30544     00640           JP PR10
                00650 ;
                00660 ;         P - PROGRAM
                00670 :
445F CD0B02     00680 PCMD      CALL G3N          ;*ADDRESS CHANGE MODEL III*
4462 ED433F47   00690           LD (NBYTES),BC    ;SAVE ARGUMENTS
4466 ED534147   00700           LD (PROMA),DE
446A 224347     00710           LD (MEMA),HL
                00720 ;
                00730 ;         CHECK IF IT'S DIRTY
                00740 ;
                00750 :         BC = #OF BYTES TO CHECK
                00760 ;         HL = MEMORY TO PROGRAM
                00770 ;         DE = OFFSET ADDRESS
                00780 :
446D CDF845     00790           CALL SR           ;SET UP TO READ
4470 010008     00800           LD BC,0800H       ;CHECK ENTIRE EPROM
4473 7B         00810 P10       LD A,E            ;LOW ADDRESS
4474 D38D       00820           OUT (PB),A
4476 7A         00830           LD A,D            ;HIGH ADDRESS AND CONTROL
4477 F670       00840           OR 070H
4479 D38E       00850           OUT (PC),A
447B C5         00860           PUSH BC
447C C1         00870           POP BC            ;SHORT DELAY
447D DB8C       00880           IN A,(PA)         ;GET DATA
447F FEFF       00890           CP 0FFH
4481 C29444     00900           JP NZ,P20         ;IT'S DIRTY
4484 23         00910           INC HL
4485 0B         00920           DEC BC
4486 78         00930           LD A,B
4487 B1         00940           OR C
4488 C27344     00950           JP NZ,P10
448B 21E546     00960           LD HL,MESB
448E CD8D01     00970           CALL PMSG
4491 C3F944     00980           JP P80            ;IT'S CLEAN
4494 218146     00990 P20       LD HL,MES5
4497 CD8D01     01000           CALL PMSG         ;PRINT 'DIRTY PROM.'
                01010 ;
                01020 ;         DETERMINE WHETHER IT CAN BE PROGRAMMED
                01030 ;         IF ATTEMPT IS MADE TO CHANGE A 0 TO 1,
                01040 ;         IT WON'T WORK
                01050 ;
449A ED5B4147   01060           LD DE,(PROMA)
449E 2A4347     01070           LD HL,(MEMA)
44A1 ED4B3F47   01080           LD BC,(NBYTES)
44A5 CDF845     01090           CALL SR           ;SET UP TO READ
44A8 7B         01100 P30       LD A,E
44A9 D38D       01110           OUT (PB),A
44AB 7A         01120           LD A,D
44AC F670       01130           OR 070H
44AE D38E       01140           OUT (PC),A
```

```
44B0 C5          01150          PUSH BC
44B1 C1          01160          POP BC           ;SHORT DELAY
44B2 DB8C        01170          IN A,(PA)
44B4 2F          01180          CPL
44B5 A6          01190          AND (HL)
44B6 C2F044      01200          JP NZ,PNO        ;IF NZ IT WON'T WORK
44B9 13          01210          INC DE
44BA 23          01220          INC HL
44BB 0B          01230          DEC BC
44BC 78          01240          LD A,B
44BD B1          01250          OR C
44BE C2A844      01260          JP NZ,P30
44C1 21A446      01270          LD HL,MES7       ;PRINT 'IT COULD WORK'
44C4 CD8D01      01280 P50      CALL PMSG
44C7 21B546      01290          LD HL,MES8       ;PRINT 'CONTINUE?'
44CA CD8D01      01300          CALL PMSG
44CD 214547      01310 P60      LD HL,BUFFER
44D0 CD2701      01320          CALL BUF
44D3 214547      01330          LD HL,BUFFER
44D6 23          01340 P70      INC HL           ;CHECK IF YES OR NO TYPED IN
44D7 7E          01350          LD A,(HL)
44D8 FE59        01360          CP 'Y'           ;IF YES, CONTINUE
44DA CAF944      01370          JP Z,P80
44DD FE4E        01380          CP 'N'
44DF CA0544      01390          JP Z,PR10        ;IF NO, GO BACK TO BEGINNING
44E2 FE0D        01400          CP 0DH
44E4 C2D644      01410          JP NZ,P70
44E7 21C046      01420          LD HL,MES9       ;PRINT 'ANSWER YES OR NO'
44EA CD8D01      01430          CALL PMSG
44ED C3CD44      01440          JP P60
44F0 219046      01450 PNO      LD HL,MES6       ;PRINT 'IT WON'T WORK'
44F3 CD8D01      01460          CALL PMSG
44F6 C30544      01470          JP PR10          ;GO BACK FOR NEW COMMAND
                 01480 ;
                 01490 ;        ATTEMPT TO PROGRAM THE 2716 EPROM
                 01500 ;
44F9 210947      01510 P80      LD HL,MESC       ;PRINT'WAIT FOR PROGRAMMING
                 01520                           ; MAX TIME 2 MINUTES'
44FC CD8D01      01530          CALL PMSG
44FF ED4B3F47    01540          LD BC,(NBYTES)
4503 2A4347      01550          LD HL,(MEMA)
4506 ED5B4147    01560          LD DE,(PROMA)    ;OFFSET
450A CDB645      01570          CALL PROG        ;PROGRAM IT
450D 2A4347      01580 P90      LD HL,(MEMA)
4510 ED5B4147    01590          LD DE,(PROMA)
4514 ED4B3F47    01600          LD BC,(NBYTES)
4518 CD3745      01610          CALL VERIFY
451B C30544      01620          JP PR10
                 01630 ;
                 01640 ;        READ - READ PROM INTO MEMORY
                 01650 ;
                 01660 ;        BC = # BYTES TO READ
                 01670 ;        DE = PROM ADDRESS
                 01680 ;        HL = MEMORY ADDRESS
                 01690 ;
451E CDF845      01700 READ     CALL SR          ;SET UP CHIP TO READ
4521 7B          01710 RD10     LD A,E
```

```
4522 D38D      01720          OUT (PB),A      ;OUTPUT A0-A7
4524 7A        01730          LD A,D
4525 F670      01740          OR 070H
4527 D38E      01750          OUT (PC),A      ;OUTPUT A8-A10
4529 C5        01760          PUSH BC
452A C1        01770          POP BC          ;SHORT DELAY
452B DB8C      01780          IN A,(PA)       ;GET DATA BYTES
452D 77        01790          LD (HL),A       ;STORE IT IN MEMORY
452E 13        01800          INC DE
452F 23        01810          INC HL          ;GET READY TO READ NEXT BYTE
4530 0B        01820          DEC BC
4531 78        01830          LD A,B
4532 B1        01840          OR C
4533 C22145    01850          JP NZ,RD10
4536 C9        01860          RET
               01870 ;
               01880 ;        VERIFY - VERIFY MEMORY AGAINST PROM MEMORY
               01890 ;
               01900 ;        HL = MEMORY ADDRESS
               01910 ;        DE = PROM MEMORY ADDRESS
               01920 ;        BC = # BYTES TO VERIFY
               01930 ;
4537 E5        01940 VERIFY   PUSH HL
4538 210000    01950          LD HL,0000H     ;ERRS = 0
453B 223D47    01960          LD (ERRS),HL
453E E1        01970          POP HL
               01980 ;
453F CDF845    01990          CALL SR         ;SET UP CHIP TO READ
4542 7B        02000 VE10     LD A,E
4543 D38D      02010          OUT (PB),A      ;OUTPUT A0-A7
4545 7A        02020          LD A,D
4546 F670      02030          OR 070H
4548 D38E      02040          OUT (PC),A      ;OUTPUT A8-A10
454A C5        02050          PUSH BC
454B C1        02060          POP BC
454C C5        02070          PUSH BC
454D C1        02080          POP BC
454E DB8C      02090          IN A,(PA)
4550 BE        02100          CP (HL)
4551 CA9245    02110          JP Z,VE100      ;IF MATCH, INCREMENT POINTERS
               02120 ;
               02130 ;        THERE IS A DISCREPANCY
               02140 ;        WRITE 'MEMORY: XXXX XX   PROM: XXXX XX'
               02150 ;
4554 C5        02160          PUSH BC
4555 E5        02170          PUSH HL
4556 D5        02180          PUSH DE
4557 F5        02190          PUSH AF
4558 115446    02200          LD DE,MES31     ;FORMAT MEMORY ADDRESS
455B 7C        02210          LD A,H
455C CD0346    02220          CALL I2AH
455F 13        02230          INC DE
4560 13        02240          INC DE
4561 7D        02250          LD A,L
4562 CD0346    02260          CALL I2AH
4565 115946    02270          LD DE,MES32     ;FORMAT MEMORY CONTENT
4568 7E        02280          LD A,(HL)
```

```
4569 CD0346    02290          CALL I2AH
456C 116A46    02300          LD DE,MES34      ;FORMAT PROM CONTENTS
456F F1        02310          POP AF
4570 CD0346    02320          CALL I2AH
4573 116546    02330          LD DE,MES33
4576 E1        02340          POP HL           ;GET PROM MEMORY ADDRESS
4577 E5        02350          PUSH HL
4578 7C        02360          LD A,H
4579 CD0346    02370          CALL I2AH        ;FORMAT PROM MEMORY ADDRESS
457C 13        02380          INC DE
457D 13        02390          INC DE
457E 7D        02400          LD A,L
457F CD0346    02410          CALL I2AH
4582 2A3D47    02420          LD HL,(ERRS)     ;ERRS = ERRS + 1
4585 23        02430          INC HL
4586 223D47    02440          LD (ERRS),HL
4589 214B46    02450          LD HL,MES3
458C CD8D01    02460          CALL PMSG        ;PRINT THE DISCREPANCY
458F D1        02470          POP DE
4590 E1        02480          POP HL
4591 C1        02490          POP BC
               02500 ;
4592 13        02510 VE100    INC DE
4593 23        02520          INC HL
4594 0B        02530          DEC BC
4595 3A8038    02540          LD A,(3880H)     ;SHIFT FOR ESCAPE
4598 B7        02550          OR A
4599 C0        02560          RET NZ           ;ABORT
459A 78        02570          LD A,B
459B B1        02580          OR C
459C C24245    02590          JP NZ,VE10
               02600 ;
459F 2A3D47    02610          LD HL,(ERRS)     ;FORMAT ERRORS
45A2 116E46    02620          LD DE,MES41
45A5 7C        02630          LD A,H
45A6 CD0346    02640          CALL I2AH
45A9 13        02650          INC DE
45AA 13        02660          INC DE
45AB 7D        02670          LD A,L
45AC CD0346    02680          CALL I2AH
45AF 216D46    02690          LD HL,MES4       ;PRINT 'XXXX DISCREPANCIES'
45B2 CD8D01    02700          CALL PMSG
45B5 C9        02710          RET
               02720 ;
               02730 ;        PROG - WRITE MEMORY INTO PROM
               02740 ;
               02750 ;        LONGEST TIME APROX. 2 MIN.
               02760 ;
45B6 3E80      02770 PROG     LD A,80H         ;MODE 0, ALL PORTS = OUTPUT
45B8 D38F      02780          OUT (PO),A
45BA 2A4347    02790 PG10     LD HL,(MEMA)     ;HL = ADDR OF FIRST WORD IN BUFFER
45BD ED5B4147  02800          LD DE,(PROMA)    ;OFFSET
45C1 ED4B3F47  02810          LD BC,(NBYTES)
45C5 7E        02820 PG20     LD A,(HL)
45C6 D38C      02830          OUT (PA),A       ;OUTPUT D0-D7
45C8 7B        02840          LD A,E
45C9 D38D      02850          OUT (PB),A       ;OUTPUT A0-A7
```

```
45CB 7A       02860          LD A,D
45CC F638     02870          OR 038H          ;TURN ON +25 AND OUT ENABLE (20)
45CE D38E     02880          OUT (PC),A       ;OUTPUT A8-A10
45D0 C5       02890          PUSH BC          ;SAVE
45D1 F5       02900          PUSH AF          ;SAVE
45D2 D38E     02910          OUT (PC),A       ;PROGRAMMING STARTS
45D4 EE80     02920          XOR 80H
45D6 D38E     02930          OUT (PC),A       ;PULSE IT
              02940 ;
              02950 ;        CRITICAL TIME - MODEL DEPENDENT
              02960 :
45D8 01800D   02970          LD BC,0D80H      :** 0F6EH MODEL III **
              02980                           ; 50 MILLISECOND DELAY
45DB 0B       02990 PG30     DEC BC           ;WAIT
45DC 78       03000          LD A,B
45DD B1       03010          OR C
45DE 20FB     03020          JR NZ,PG30
45E0 F1       03030          POP AF
45E1 C1       03040          POP BC           ;REGISTERS RESTORED
45E2 EE80     03050          XOR 80H
45E4 D38E     03060          OUT (PC),A       :END OF PULSE
45E6 F670     03070          OR 070H          :ALL VOLTAGES OFF
45E8 D38E     03080          OUT (PC),A
45EA 3A8038   03090          LD A,(3880H)     ;SHIFT FOR ESCAPE
45ED B7       03100          OR A
45EE C0       03110          RET NZ           ;ABORT
45EF 0B       03120          DEC BC
45F0 78       03130          LD A,B
45F1 B1       03140          OR C             ;SET 0 FLAG FOR END
45F2 23       03150          INC HL           ;ADJUST POINTERS
45F3 13       03160          INC DE           ;NEXT LOCATION
45F4 C2C545   03170          JP NZ,PG20       ;IF NOT FINSHED, DO NEXT
45F7 C9       03180          RET
              03190 ;
              03200 ;
              03210 ;
              03220 ;        SR - SET UP CHIP TO READ
              03230 :
45F8 3E90     03240 SR       LD A,90H
45FA D38F     03250          OUT (P0),A       ;MODE 0, PORT A = INPUT
              03260                           ;PORT B AND C = OUTPUT
45FC 3E70     03270          LD A,70H
45FE D38E     03280          OUT (PC),A
4600 C5       03290          PUSH BC
4601 C1       03300          POP BC           ;SHORT DELAY
4602 C9       03310          RET
              03320 ;
              03330 ;        I2AH - INTEGER TO ASCII HEX
              03340 ;
              03350 ;        A = INTEGER TO CONVERT
              03360 ;        DE = ADDRESS OF ASCII BUFFER
              03370 ;        ONLY THE FLAG REGISTER IS ALTERED
              03380 ;
4603 C5       03390 I2AH     PUSH BC
4604 13       03400          INC DE           ;CONVERT LAST NIBBLE FIRST
4605 47       03410          LD B,A           ;SAVE THE INTEGER
4606 E60F     03420          AND 0FH
```

```
4608 FE0A      03430          CP 0AH
460A FA0F46    03440          JP M,IH10
460D C607      03450          ADD A,07          ;ADD 7 FOR A-F
460F C630      03460 IH10     ADD A,30H
4611 12        03470          LD (DE),A
4612 1B        03480          DEC DE            ;NOW CONVERT FIRST NIBBLE
4613 78        03490          LD A,B
4614 07        03500          RLCA              ;PUT LEFT NIBBLE
4615 07        03510          RLCA              ; IN RIGHT NIBBLE
4616 07        03520          RLCA
4617 07        03530          RLCA
4618 E60F      03540          AND 0FH
461A FE0A      03550          CP 0AH
461C FA2146    03560          JP M,IH20
461F C607      03570          ADD A,7           ;ADD 7 FOR A-F
4621 C630      03580 IH20     ADD A,30H
4623 12        03590          LD (DE),A
4624 78        03600          LD A,B            ;RESTORE REGISTERS
4625 C1        03610          POP BC
4626 C9        03620          RET
               03630 ;
4627 0D        03640 MES1     DEFB 0DH
4628 0D        03650          DEFB 0DH          ;SPACE TO AID READ
4629 32        03660          DEFM '2716 EPROM PROGRAMMING PROGRAM'
4647 01        03670          DEFB 01H
4648 0D        03680 MES2     DEFB 0DH
4649 3E        03690          DEFM '>'
464A 01        03700          DEFB 01H
464B 0D        03710 MES3     DEFB 0DH
464C 4D        03720          DEFM 'MEMORY: '
0004           03730 MES31    DEFS 4
4658 20        03740          DEFM ' '
0002           03750 MES32    DEFS 2
465B 20        03760          DEFM '    PROM: '
0004           03770 MES33    DEFS 4
4669 20        03780          DEFM ' '
0002           03790 MES34    DEFS 2
466C 01        03800          DEFB 01H
466D 0D        03810 MES4     DEFB 0DH
0004           03820 MES41    DEFS 4
4672 20        03830          DEFM ' DISCREPANCIES'
4680 01        03840          DEFB 01H
4681 0D        03850 MES5     DEFB 0DH
4682 44        03860          DEFM 'DIRTY PROM.  '
468F 01        03870          DEFB 01H
4690 49        03880 MES6     DEFM 'IT WILL NOT WORK.  '
46A3 01        03890          DEFB 01H
46A4 49        03900 MES7     DEFM 'IT COULD WORK.  '
46B4 01        03910          DEFB 01H
46B5 43        03920 MES8     DEFM 'CONTINUE? '
46BF 01        03930          DEFB 01H
46C0 0D        03940 MES9     DEFB 0DH
46C1 41        03950          DEFM 'ANSWER YES OR NO: '
46D3 01        03960          DEFB 01H
46D4 0D        03970 MESA     DEFB 0DH
46D5 55        03980          DEFM 'UNKNOWN COMMAND'
46E4 01        03990          DEFB 01H
```

```
46E5 0D        04000 MESB    DEFB 0DH
46E6 45        04010         DEFM 'EPROM CLEAN IN LOCATIONS REQUESTED'
4708 01        04020         DEFB 01H
4709 0D        04030 MESC    DEFB 0DH
470A 57        04040         DEFM 'WAIT FOR PROGRAMMING'
471E 0D        04050         DEFB 0DH
471F 4D        04060         DEFM 'MAXIMUM TIME ABOUT 2 MINUTES'
473B 01        04070         DEFB 01H
               04080 ;
0001           04090 SPFLAG  DEFS 1
0002           04100 ERRS    DEFS 2
0002           04110 NBYTES  DEFS 2
0002           04120 PROMA   DEFS 2
0002           04130 MEMA    DEFS 2
0043           04140 BUFFER  DEFS 67
               04150 ;
               04160 ;       MONITOR SUBROUTINES
               04170 ;
0127           04180 BUF     EQU 0127H       ;MODEL III NO CHANGE
018D           04190 PMSG    EQU 018DH       ;MODEL III NO CHANGE
020B           04200 G3N     EQU 020BH       ;MODEL III EQU 01FAH
008C           04210 PA      EQU 8CH
008D           04220 PB      EQU 8DH
008E           04230 PC      EQU 8EH
008F           04240 P0      EQU 8FH
0000           04250         END
00000 TOTAL ERRORS


BUF     0127 04180    00240 01320
BUFFER 4745 04140    00220 01310 01330
ERRS    473D 04100    01960 02420 02440 02610
EXIT    4448 00500    00440
G3N     020B 04200    00560 00620 00680
I2AH    4603 03390    02220 02260 02290 02320 02370 02410 02640
                      02680
IH10    460F 03460    03440
IH20    4621 03580    03560
MEMA    4743 04130    00710 01070 01550 01580 02790
MES1    4627 03640    00180
MES2    4648 03680    00200
MES3    464B 03710    02450
MES31   4654 03730    02200
MES32   4659 03750    02270
MES33   4665 03770    02330
MES34   466A 03790    02300
MES4    466D 03810    02690
MES41   466E 03820    02620
MES5    4681 03850    00990
MES6    4690 03880    01450
MES7    46A4 03900    01270
MES8    46B5 03920    01290
MES9    46C0 03940    01420
MESA    46D4 03970    00470
MESB    46E5 04000    00960
MESC    4709 04030    01510
NBYTES 473F 04110    00690 01080 01540 01600 02810
```

```
P0      008F 04240   02780 03250
P10     4473 00810   00950
P20     4494 00990   00900
P30     44A8 01100   01260
P50     44C4 01280
P60     44CD 01310   01440
P70     44D6 01340   01410
P80     44F9 01510   00980 01370
P90     450D 01580
PA      008C 04210   00880 01170 01780 02090 02830
PB      008D 04220   00820 01110 01720 02010 02850
PC      008E 04230   00850 01140 01750 02040 02880 02910 02930
                     03060 03080 03280
PCMD    445F 00680   00420
PG10    45BA 02790
PG20    45C5 02820   03170
PG30    45DB 02990   03020
PMSG    018D 04190   00190 00210 00480 00970 01000 01280 01300
                     01430 01460 01530 02460 02700
PNO     44F0 01450   01200
PR10    4405 00180   00490 00580 00640 01390 01470 01620
PR20    4422 00350   00460
PROG    45B6 02770   01570
PROMA   4741 04120   00700 01060 01560 01590 02800
RCMD    444D 00560   00380
RD10    4521 01710   01850
READ    451E 01700   00570
SPFLAG  473C 04090
SR      45F8 03240   00170 00790 01090 01700 01990
VCMD    4456 00620   00400
VE10    4542 02000   02590
VE100   4592 02510   02110
VERIFY  4537 01940   00630 01610
```

APPENDIX H

Source Listing for the 8755 EPROM Programmer

```
                00010 ;       PROM - BURN A 8755
                00020 ;       DISK NAME  P8755/ASM
                00030 ;       FOR MODIFIED TRS-80 SYSTEMS
                00040 ;
                00050 ;       COMMANDS:
                00060 ;       READ:     R  MEMORY ADDR., # BYTES, PROM ADDR.
                00070 ;       VERIFY:   V       "       ,     "   ,      "
                00080 ;       PROGRAM:  P       "       ,     "   ,      "
                00090 ;       EXIT:     E
                00100 ;
                00110 ;       SUBROUTINES USED FROM MONITOR:
                00120 ;       BUF, G3N, PMSG
                00130 ;
                00140 ;       PORT ASSIGNMENTS
                00150 ;
                00160 ;              PORT A
                00170 ;          ADDRESS 0-7 DATA 0-7
                00180 ;
                00190 ;              PORT B
                00200 ;E0 - ALE      B1 - IO/M       B2 - PROG/CE1   E3 - CE2
                00210 ;B4 - /IOW     B5 - /IOR       B6 - CLK        E7 - /RD
                00220 ;
                00230 ;              PORT C
                00240 ;B0 - A8       B1 - A9         B2 - A10        B3 - XX
                00250 ;B4 - XX       B5 - XX         B6 - PROG PULSE B7 - XX
                00260 ;
4400            00270         ORG 04400H
                00280 ;
4400 FDE5       00290         PUSH IY          ;SAVE MONITOR POINTER
4402 CDFA45     00300         CALL SO          ;SET UP AND REMOVE VOLTAGES
4405 213646     00310 PR10    LD HL,MES1       ;PRINT 'PROM PROGRAMMING PROGRAM'
4408 CD8D01     00320         CALL PMSG
440B 215746     00330         LD HL,MES2       ;PRINT PROMPT (>)
440E CD8D01     00340         CALL PMSG
4411 215447     00350         LD HL,BUFFER
4414 E5         00360         PUSH HL
4415 CD2701     00370         CALL BUF         ;GET COMMAND BUFFER
4418 3620       00380         LD (HL),20H      ;PUT SPACE
441A 23         00390         INC HL
441B 3630       00400         LD (HL),30H      ;IN CASE THIRD ARGUMENT NOT ENT
441D 23         00410         INC HL
441E 362C       00420         LD (HL),','      ;END
4420 FDE1       00430         POP IY
                00440 ;
                00450 ;       BRANCH TO CORRECT SECTION OF PROGRAM DEPENDING
                00460 ;       ON COMMAND ENTERED
                00470 ;
4422 FD23       00480 PR20    INC IY
4424 FD7E00     00490         LD A,(IY+0)
4427 FE52       00500         CP 'R'
4429 CA4D44     00510         JP Z,RCMD
442C FE56       00520         CP 'V'
442E CA5644     00530         JP Z,VCMD
4431 FE50       00540         CP 'P'
4433 CA5F44     00550         JP Z,PCMD
4436 FE45       00560         CP 'E'
4438 280E       00570         JR Z,EXIT        ;IF EXIT, RETURN TO MONITOR
```

```
443A FE2C      00580          CP ','           ;IF COMMA, PRINT 'UNKNOWN COMMAND'
443C C22244    00590          JP NZ,PR20
443F 21E346    00600          LD HL,MESA
4442 CD8D01    00610          CALL PMSG
4445 C30544    00620          JP PR10
4448 FDE1      00630 EXIT     POP IY
444A FD23      00640          INC IY           ;ADJUST BUFFER
444C C9        00650          RET
               00660 ;
               00670 ;        R - READ
               00680 ;
444D CD0B02    00690 RCMD     CALL G3N         ;GET ARGUMENTS
4450 CD2445    00700          CALL READ        ;SUBROUTINE READ DOES THE WORK
4453 C30544    00710          JP PR10
               00720 ;
               00730 ;        V - VERIFY
               00740 ;
4456 CD0B02    00750 VCMD     CALL G3N         ;GET ARGUMENTS
4459 CD4045    00760          CALL VERIFY      ;SUBROUTINE VERIFY DOES THE WORK
445C C30544    00770          JP PR10
               00780 ;
               00790 ;        P - PROGRAM
               00800 ;
445F CD0B02    00810 PCMD     CALL G3N
4462 ED434E47  00820          LD (NBYTES),BC   ;SAVE ARGUMENTS
4466 ED535047  00830          LD (PROMA),DE
446A 225247    00840          LD (MEMA),HL
               00850 ;
               00860 ;        CHECK IF IT'S DIRTY
               00870 ;
               00880 ;        BC = #OF BYTES TO CHECK
               00890 ;        HL = MEMORY TO PROGRAM
               00900 ;        DE = OFFSET ADDRESS
446D 010008    00910          LD BC,0800H      ;CHECK ENTIRE EPROM
4470 CDFA45    00920 P10      CALL SO          ;SET UP AS OUTPUT
4473 CD0146    00930          CALL LADR        ;LATCH ADDRESS IN 8755
4476 3E90      00940          LD A,90H         ;CHANGE A TO INPUT PORT
4478 D38F      00950          OUT (P0),A       ;SEND TO CONTROL
447A 3E38      00960          LD A,RD          ;SET RD LOW
447C D38D      00970          OUT (PB),A
447E C5        00980          PUSH BC
447F C1        00990          POP BC           ;SHORT DELAY
4480 DB8C      01000          IN A,(PA)        ;GET DATA
4482 FEFF      01010          CP 0FFH
4484 C29744    01020          JP NZ,P20        ;IT'S DIRTY
4487 23        01030          INC HL
4488 0B        01040          DEC BC
4489 78        01050          LD A,B
448A B1        01060          OR C
448B C27044    01070          JP NZ,P10
448E 21F446    01080          LD HL,MESB
4491 CD8D01    01090          CALL PMSG
4494 C3FF44    01100          JP P80           ;IT'S CLEAN
4497 219046    01110 P20      LD HL,MES5
449A CD8D01    01120          CALL PMSG        ;PRINT 'DIRTY PROM.'
               01130 ;
               01140 ;        DETERMINE WHETHER IT CAN BE PROGRAMMED
```

```
                01150 ;         IF ATTEMPT 1S MADE TO CHANGE A 0 TO 1,
                01160 ;         1T WON'T WORK
                01170 ;
449D ED5B5047   01180           LD DE,(PROMA)
44A1 2A5247     01190           LD HL,(MEMA)
44A4 ED4B4E47   01200           LD BC,(NBYTES)
44A8 CDFA45     01210 P30       CALL SO             ;SET UP AS ALL OUTPUT
44AB CD0146     01220           CALL LADR
44AE 3E90       01230           LD A,90H            ;CHANGE TO INPUT PORT
44B0 D38F       01240           OUT (P0),A
44B2 3E38       01250           LD A,RD             ;SET TO READ
44B4 D38D       01260           OUT (PB),A
44B6 C5         01270           PUSH BC
44B7 C1         01280           POP BC              ;SHORT DELAY
44B8 DB8C       01290           1N A,(PA)
44BA 2F         01300           CPL
44BB A6         01310           AND (HL)
44BC C2F644     01320           JP NZ,PNO           ;IF NZ IT WON'T WORK
44BF 13         01330           1NC DE
44C0 23         01340           1NC HL
44C1 0B         01350           DEC BC
44C2 78         01360           LD A,B
44C3 B1         01370           OR C
44C4 C2A844     01380           JP NZ,P30
44C7 21B346     01390           LD HL,MES7          ;PR1NT 'IT COULD WORK'
44CA CD8D01     01400 P50       CALL PMSG
44CD 21C446     01410           LD HL,MES8          ;PR1NT 'CONT1NUE?'
44D0 CD8D01     01420           CALL PMSG
44D3 215447     01430 P60       LD HL,BUFFER
44D6 CD2701     01440           CALL BUF
44D9 215447     01450           LD HL,BUFFER
44DC 23         01460 P70       1NC HL              ;CHECK 1F YES OR NO TYPED 1N
44DD 7E         01470           LD A,(HL)
44DE FE59       01480           CP 'Y'              ;IF YES, CONT1NUE
44E0 CAFF44     01490           JP Z,P80
44E3 FE4E       01500           CP 'N'
44E5 CA0544     01510           JP Z,PR10           ;IF NO, GO BACK TO BEGINNING
44E8 FE0D       01520           CP 0DH
44EA C2DC44     01530           JP NZ,P70
44ED 21CF46     01540           LD HL,MES9          :PRINT 'ANSWER YES OR NO'
44F0 CD8D01     01550           CALL PMSG
44F3 C3D344     01560           JP P60
44F6 219F46     01570 PNO       LD HL,MES6          :PRINT '1T WON'T WORK'
44F9 CD8D01     01580           CALL PMSG
44FC C30544     01590           JP PR10             ;GO BACK FOR NEW COMMAND
                01600 ;
                01610 ;         ATTEMPT TO PROGRAM THE 8755 EPROM
                01620 ;
44FF 211847     01630 P80       LD HL,MESC          ;PR1NT'WA1T FOR PROGRAMM1NG
                01640                               : MAX T1ME 2 M1NUTES'
4502 CD8D01     01650           CALL PMSG
4505 ED4B4E47   01660           LD BC,(NBYTES)
4509 2A5247     01670           LD HL,(MEMA)
450C ED5B5047   01680           LD DE,(PROMA)       ;OFFSET
4510 CDC045     01690           CALL PROG           ;PROGRAM IT
4513 2A5247     01700 P90       LD HL,(MEMA)
4516 ED5B5047   01710           LD DE,(PROMA)
```

```
451A ED4B4E47 01720          LD BC,(NBYTES)
451E CD4045   01730          CALL VERIFY
4521 C30544   01740          JP PR10
              01750 ;
              01760 ;        READ - READ PROM INTO MEMORY
              01770 ;
              01780 ;        BC = # BYTES TO READ
              01790 ;        DE = PROM ADDRESS
              01800 ;        HL = MEMORY ADDRESS
              01810 ;
4524 CDFA45   01820 READ     CALL SO          ;SET UP CHIP AS OUTPUT
4527 CD0146   01830          CALL LADR
452A 3E90     01840          LD A,90H         ;DATA IN
452C D38F     01850          OUT (P0),A
452E 3E38     01860          LD A,RD          ;READ 8755
4530 D38D     01870          OUT (PB),A
4532 C5       01880          PUSH BC
4533 C1       01890          POP BC           ;SHORT DELAY
4534 DB8C     01900          IN A,(PA)        ;GET DATA BYTES
4536 77       01910          LD (HL),A        ;STORE IT IN MEMORY
4537 13       01920          INC DE
4538 23       01930          INC HL           ;GET READY TO READ NEXT BYTE
4539 0B       01940          DEC BC
453A 78       01950          LD A,B
453B B1       01960          OR C
453C C22445   01970          JP NZ,READ
453F C9       01980          RET
              01990 ;
              02000 ;        VERIFY - VERIFY MEMORY AGAINST PROM MEMORY
              02010 ;
              02020 ;        HL = MEMORY ADDRESS
              02030 ;        DE = PROM MEMORY ADDRESS
              02040 ;        BC = # BYTES TO VERIFY
              02050 ;
4540 E5       02060 VERIFY   PUSH HL
4541 210000   02070          LD HL,0000H      ;ERRS = 0
4544 224C47   02080          LD (ERRS),HL
4547 E1       02090          POP HL
              02100 ;
4548 CDFA45   02110 VE10     CALL SO          ;SET UP CHIP TO ALL OUTPUT
454B CD0146   02120          CALL LADR
454E 3E90     02130          LD A,90H
4550 D38F     02140          OUT (P0),A       ;DATA IN
4552 3E38     02150          LD A,RD          ;SET UP TO READ
4554 D38D     02160          OUT (PB),A
4556 C5       02170          PUSH BC
4557 C1       02180          POP BC
4558 DB8C     02190          IN A,(PA)
455A BE       02200          CP (HL)
455B CAA145   02210          JP Z,VE100       ;IF MATCH, INCREMENT POINTERS
              02220 ;
              02230 ;        THERE IS A DISCREPANCY
              02240 ;        WRITE 'MEMORY: XXXX XX   PROM: XXXX XX'
              02250 ;
455E C5       02260          PUSH BC
455F E5       02270          PUSH HL
4560 D5       02280          PUSH DE
```

```
4561 F5       02290          PUSH AF
4562 116346   02300          LD DE,MES31     ;FORMAT MEMORY ADDRESS
4565 7C       02310          LD A,H
4566 CD1246   02320          CALL I2AH
4569 13       02330          INC DE
456A 13       02340          INC DE
456B 7D       02350          LD A,L
456C CD1246   02360          CALL I2AH
456F 116846   02370          LD DE,MES32     ;FORMAT MEMORY CONTENT
4572 7E       02380          LD A,(HL)
4573 CD1246   02390          CALL I2AH
4576 117946   02400          LD DE,MES34     ;FORMAT PROM CONTENTS
4579 F1       02410          POP AF
457A CD1246   02420          CALL I2AH
457D 117446   02430          LD DE,MES33
4580 E1       02440          POP HL          ;GET PROM MEMORY ADDRESS
4581 E5       02450          PUSH HL
4582 7C       02460          LD A,H
4583 CD1246   02470          CALL I2AH       ;FORMAT PROM MEMORY ADDRESS
4586 13       02480          INC DE
4587 13       02490          INC DE
4588 7D       02500          LD A,L
4589 CD1246   02510          CALL I2AH
458C 2A4C47   02520          LD HL,(ERRS)    ;ERRS = ERRS + 1
458F 23       02530          INC HL
4590 224C47   02540          LD (ERRS),HL
4593 215A46   02550          LD HL,MES3
4596 CD8D01   02560          CALL PMSG       ;PRINT THE DISCREPANCY
4599 3A8038   02570          LD A,(3880H)    ;CHECK FOR SHIFT
459C B7       02580          OR A
459D C0       02590          RET NZ          ;ABORT
459E D1       02600          POP DE
459F E1       02610          POP HL
45A0 C1       02620          POP BC
              02630 ;
45A1 13       02640 VE100    INC DE
45A2 23       02650          INC HL
45A3 0B       02660          DEC BC
45A4 78       02670          LD A,B
45A5 B1       02680          OR C
45A6 C24845   02690          JP NZ,VE10
              02700 ;
45A9 2A4C47   02710          LD HL,(ERRS)    ;FORMAT ERRS
45AC 117D46   02720          LD DE,MES41
45AF 7C       02730          LD A,H
45B0 CD1246   02740          CALL I2AH
45B3 13       02750          INC DE
45B4 13       02760          INC DE
45B5 7D       02770          LD A,L
45B6 CD1246   02780          CALL I2AH
45B9 217C46   02790          LD HL,MES4      ;PRINT 'XXXX DISCREPANCIES'
45BC CD8D01   02800          CALL PMSG
45BF C9       02810          RET
              02820 ;
              02830 ;        PROG - WRITE MEMORY INTO PROM
              02840 ;
              02850 ;        LONGEST TME APROX. 2 MIN.
```

```
                02860 ;
45C0 2A5247     02870 PROG      LD HL,(MEMA)     ;HL = ADDR OF FIRST WORD IN BUFFER
45C3 ED5B5047   02880           LD DE,(PROMA)    ;OFFSET
45C7 ED4B4E47   02890           LD BC,(NBYTES)
45CB 3A8038     02900 PG20      LD A,(3880H)     ;CHECK FOR SHIFT
45CE B7         02910           OR A
45CF C0         02920           RET NZ           ;ABORT
45D0 CDFA45     02930           CALL SO          ;ALL OUTPUT
45D3 CD0146     02940           CALL LADR
45D6 7E         02950           LD A,(HL)        ;GET DATA
45D7 D38C       02960           OUT (PA),A       ;OUTPUT DATA
45D9 3EBC       02970           LD A,PGWD        ;TURN ON CE2,PROG, & +25 OFF
45DB D38D       02980           OUT (PB),A
45DD 3ECF       02990           LD A,0CFH        ;ONLY WANT BITS 6 & 7 LOW FOR PULSE
45DF C5         03000           PUSH BC          ;SAVE
45E0 F5         03010           PUSH AF          ;SAVE
45E1 D38E       03020           OUT (PC),A       ;PROGRAMMING STARTS
                03030 ;
                03040 ;         CRITICAL TIME - MODEL DEPENDENT
                03050 ;
45E3 01800D     03060           LD BC,0D80H      ;** 0F6EH MODEL II1 **
                03070                            ; 50 MILLISECOND DELAY
45E6 0B         03080 PG30      DEC BC           ;WAIT
45E7 78         03090           LD A,B
45E8 B1         03100           OR C
45E9 20FB       03110           JR NZ,PG30
45EB F1         03120           POP AF
45EC C1         03130           POP BC           ;REGISTERS RESTORED
45ED EE40       03140           XOR 40H          ;PULSE OFF
45EF D38E       03150           OUT (PC),A       ;END OF PULSE, VOLTAGES OFF
45F1 0B         03160           DEC BC
45F2 78         03170           LD A,B
45F3 B1         03180           OR C             ;SET 0 FLAG FOR END
45F4 23         03190           INC HL           ;ADJUST POINTERS
45F5 13         03200           INC DE           ;NEXT LOCATION
45F6 C2CB45     03210           JP NZ,PG20       ;IF NOT FINSHED, DO NEXT
45F9 C9         03220           RET
                03230 ;
                03240 ;
                03250 ;
                03260 ;         SO - SET UP CHIP TO OUTPUT
                03270 ;
45FA 3E80       03280 SO        LD A,80H
45FC D38F       03290           OUT (P0),A       ;MODE 0, ALL OUTPUT
45FE C5         03300           PUSH BC
45FF C1         03310           POP BC           ;SHORT DELAY
4600 C9         03320           RET
                03330 ;
                03340 ;         LATCH ADDRESS ON 8755
                03350 ;
4601 7B         03360 LADR      LD A,E           ;E HAS OFFSET LOW ADDRSS
4602 D38C       03370           OUT (PA),A       ;TO DATA ADDRESS LINES
4604 7A         03380           LD A,D           ;HIGH ADDRESS
4605 F678       03390           OR 78H           ;PROGRAM VOLTAGE OFF
4607 D38E       03400           OUT (PC),A       ;ADDRESS AND VOLTAGES SET
4609 3EB9       03410           LD A,ADDSET      ;ADDRESS SET UP
460B D38D       03420           OUT (PB),A
```

```
460D 3EB8      03430          LD A,ALEON
460F D38D      03440          OUT (PB),A      ;ADDRESS LATCHED
4611 C9        03450          RET
               03460 ;
               03470 ;        I2AH - INTEGER TO ASCII HEX
               03480 ;
               03490 ;        A = INTEGER TO CONVERT
               03500 ;        DE = ADDRESS OF ASCII BUFFER
               03510 ;        ONLY THE FLAG REGISTER IS ALTERED
               03520 ;
4612 C5        03530 I2AH     PUSH BC
4613 13        03540          INC DE          ;CONVERT LAST NIBBLE FIRST
4614 47        03550          LD B,A          ;SAVE THE INTEGER
4615 E60F      03560          AND 0FH
4617 FE0A      03570          CP 0AH
4619 FA1E46    03580          JP M,IH10
461C C607      03590          ADD A,07        ;ADD 7 FOR A-F
461E C630      03600 IH10     ADD A,30H
4620 12        03610          LD (DE),A
4621 1B        03620          DEC DE          ;NOW CONVERT FIRST NIBBLE
4622 78        03630          LD A,B
4623 07        03640          RLCA            ;PUT LEFT NIBBLE
4624 07        03650          RLCA            ; IN RIGHT NIBBLE
4625 07        03660          RLCA
4626 07        03670          RLCA
4627 E60F      03680          AND 0FH
4629 FE0A      03690          CP 0AH
462B FA3046    03700          JP M,IH20
462E C607      03710          ADD A,7         ;ADD 7 FOR A-F
4630 C630      03720 IH20     ADD A,30H
4632 12        03730          LD (DE),A
4633 78        03740          LD A,B          ;RESTORE REGISTERS
4634 C1        03750          POP BC
4635 C9        03760          RET
               03770 ;
4636 0D        03780 MES1     DEFB 0DH
4637 0D        03790          DEFB 0DH
4638 38        03800          DEFM '8875 EPROM PROGRAMMING PROGRAM'
4656 01        03810          DEFB 01H
4657 0D        03820 MES2     DEFB 0DH
4658 3E        03830          DEFM '>'
4659 01        03840          DEFB 01H
465A 0D        03850 MES3     DEFB 0DH
465B 4D        03860          DEFM 'MEMORY: '
0004           03870 MES31    DEFS 4
4667 20        03880          DEFM ' '
0002           03890 MES32    DEFS 2
466A 20        03900          DEFM '    PROM: '
0004           03910 MES33    DEFS 4
4678 20        03920          DEFM ' '
0002           03930 MES34    DEFS 2
467B 01        03940          DEFB 01H
467C 0D        03950 MES4     DEFB 0DH
0004           03960 MES41    DEFS 4
4681 20        03970          DEFM ' DISCREPANCIES'
468F 01        03980          DEFB 01H
4690 0D        03990 MES5     DEFB 0DH
```

```
4691 44        04000          DEFM 'DIRTY PROM.  '
469E 01        04010          DEFB 01H
469F 49        04020 MES6     DEFM 'IT WILL NOT WORK.  '
46B2 01        04030          DEFB 01H
46B3 49        04040 MES7     DEFM 'IT COULD WORK.  '
46C3 01        04050          DEFB 01H
46C4 43        04060 MES8     DEFM 'CONTINUE? '
46CE 01        04070          DEFB 01H
46CF 0D        04080 MES9     DEFB 0DH
46D0 41        04090          DEFM 'ANSWER YES OR NO: '
46E2 01        04100          DEFB 01H
46E3 0D        04110 MESA     DEFB 0DH
46E4 55        04120          DEFM 'UNKNOWN COMMAND'
46F3 01        04130          DEFB 01H
46F4 0D        04140 MESB     DEFB 0DH
46F5 45        04150          DEFM 'EPROM CLEAN IN LOCATIONS REQUESTED'
4717 01        04160          DEFB 01H
4718 0D        04170 MESC     DEFB 0DH
4719 57        04180          DEFM 'WAIT FOR PROGRAMMING'
472D 0D        04190          DEFB 0DH
472E 4D        04200          DEFM 'MAXIMUM TIME ABOUT 2 MINUTES'
474A 01        04210          DEFB 01H
               04220 ;
0001           04230 SPFLAG   DEFS 1
0002           04240 ERRS     DEFS 2
0002           04250 NBYTES   DEFS 2
0002           04260 PROMA    DEFS 2
0002           04270 MEMA     DEFS 2
0043           04280 BUFFER   DEFS 67
               04290 ;
               04300 ;        PROGRAMMING CONSTANTS
               04310 ;
00B8           04320 ALEON    EQU 0B8H        ;LATCHES ADDRESS LINES
00B9           04330 ADDSET   EQU 0B9H        ;SET UP ADDRESS DATA FOR LATCH
0038           04340 RD       EQU 038H        ;READ EPROM
00BC           04350 PGWD     EQU 0BCH        ;CE2 HIGH
               04360 ;
               04370 ;        MONITOR SUBROUTINES
               04380 ;
0127           04390 BUF      EQU 0127H       ;MODEL III NO CHANGE
018D           04400 PMSG     EQU 018DH       ;MODEL III NO CHANGE
020B           04410 G3N      EQU 020BH       ;MODEL III EQU 01FAH
008C           04420 PA       EQU 8CH
008D           04430 PB       EQU 8DH
008E           04440 PC       EQU 8EH
008F           04450 P0       EQU 8FH
0000           04460          END
00000 TOTAL ERRORS


ADDSET 00B9 04330   03410
ALEON  00B8 04320   03430
BUF    0127 04390   00370 01440
BUFFER 4754 04280   00350 01430 01450
ERRS   474C 04240   02080 02520 02540 02710
EXIT   4448 00630   00570
G3N    020B 04410   00690 00750 00810
```

```
I2AH    4612 03530   02320 02360 02390 02420 02470 02510 02740
                     02780
IH10    461E 03600   03580
IH20    4630 03720   03700
LADR    4601 03360   00930 01220 01830 02120 02940
MEMA    4752 04270   00840 01190 01670 01700 02870
MES1    4636 03780   00310
MES2    4657 03820   00330
MES3    465A 03850   02550
MES31   4663 03870   02300
MES32   4668 03890   02370
MES33   4674 03910   02430
MES34   4679 03930   02400
MES4    467C 03950   02790
MES41   467D 03960   02720
MES5    4690 03990   01110
MES6    469F 04020   01570
MES7    46B3 04040   01390
MES8    46C4 04060   01410
MES9    46CF 04080   01540
MESA    46E3 04110   00600
MESB    46F4 04140   01080
MESC    4718 04170   01630
NBYTES  474E 04250   00820 01200 01660 01720 02890
P0      008F 04450   00950 01240 01850 02140 03290
P10     4470 00920   01070
P20     4497 01110   01020
P30     44A8 01210   01380
P50     44CA 01400
P60     44D3 01430   01560
P70     44DC 01460   01530
P80     44FF 01630   01100 01490
P90     4513 01700
PA      008C 04420   01000 01290 01900 02190 02960 03370
PB      008D 04430   00970 01260 01870 02160 02980 03420 03440
PC      008E 04440   03020 03150 03400
PCMD    445F 00810   00550
PG20    45CB 02900   03210
PG30    45E6 03080   03110
PGWD    00BC 04350   02970
PMSG    018D 04400   00320 00340 00610 01090 01120 01400 01420
                     01550 01580 01650 02560 02800
PNO     44F6 01570   01320
PR10    4405 00310   00620 00710 00770 01510 01590 01740
PR20    4422 00480   00590
PROG    45C0 02870   01690
PROMA   4750 04260   00830 01180 01680 01710 02880
RCMD    444D 00690   00510
RD      0038 04340   00960 01250 01860 02150
READ    4524 01820   00700 01970
SO      45FA 03280   00300 00920 01210 01820 02110 02930
SPFLAG  474B 04230
VCMD    4456 00750   00530
VE10    4548 02110   02690
VE100   45A1 02640   02210
VERIFY  4540 02060   00760 01730
```

Index

D

E

F

G

H

READER SERVICE CARD

To better serve you, the reader, please take a moment to fill out this card, or a copy of it, for us. Not only will you be kept up to date on the Blacksburg Series books, but as an extra bonus, we **will randomly select five cards every month, from all of the cards sent to us during the previous month. The names that are drawn will win, absolutely free, a book from the Blacksburg Continuing Education Series.** Therefore, make sure to indicate your choice in the space provided below. For a complete listing of all the books to choose from, refer to the inside front cover of this book. Please, one card per person. Give everyone a chance.

In order to find out who has won a book in your area, call (703) 953-1861 anytime during the night or weekend. When you do call, an answering machine will let you know the monthly winners. Too good to be true? Just give us a call. Good luck.

If I win, please send me a copy of:

_____

I understand that this book will be sent to me absolutely free, if my card is selected.

For our information, how about telling us a little about yourself. We are interested in your occupation, how and where you normally purchase books and the books that you would like to see in the Blacksburg Series. We are also interested in finding authors for the series, so if you have a book idea, write to The Blacksburg Group, Inc., P.O. Box 242, Blacksburg, VA 24060 and ask for an Author Packet. We are also interested in TRS-80, APPLE, OSI and PET BASIC programs.

My occupation is _____
I buy books through/from _____
Would you buy books through the mail? _____
I'd like to see a book about _____
Name _____
Address _____
City _____
State _____ Zip _____

MAIL TO: BOOKS, BOX 715, BLACKSBURG, VA 24060
!!!!!PLEASE PRINT!!!!!

The Blacksburg Group

According to Business Week magazine (Technology July 6, 1976) large scale integrated circuits or LSI "chips" are creating a second industrial revolution that will quickly involve us all. The speed of the developments in this area is breathtaking and it becomes more and more difficult to keep up with the rapid advances that are being made. It is also becoming difficult for newcomers to "get on board."

It has been our objective, as The Blacksburg Group, to develop timely and effective educational materials that will permit students, engineers, scientists, technicians and others to quickly learn how to use new technologies and electronic techniques. We continue to do this through several means, textbooks, short courses, seminars and through the development of special electronic devices and training aids.

Our group members make their home in Blacksburg, found in the Appalachian Mountains of southwestern Virginia. While we didn't actively start our group collaboration until the Spring of 1974, members of our group have been involved in digital electronics, minicomputers and microcomputers for some time.

Some of our past experiences and on-going efforts include the following:

–The design and development of what is considered to be the first popular hobbyist computer. The Mark-8 was featured in Radio-Electronics magazine in 1974. We have also designed several 8080-based computers, including the MMD-1 system. Our most recent computer is an 8085-based computer for educational use, and for use in small controllers.

—The Blacksburg Continuing Education Series™ covers subjects ranging from basic electronics through microcomputers, operational amplifiers, and active filters. Test experiments and examples have been provided in each book. We are strong believers in the use of detailed experiments and examples to reinforce basic concepts. This series originally started as our Bugbook series and many titles are now being translated into Chinese, Japanese, German and Italian.

–We have pioneered the use of small, self-contained computers in hands-on courses for microcomputer users. Many of our designs have evolved into commercial products that are marketed by E&L Instruments and PACCOM, and are available from Group Technology, Ltd., Check, VA 24072.

–Our short courses and seminar programs have been presented throughout the world. Programs are offered by The Blacksburg Group, and by the Virginia Polytechnic Institute Extension Division. Each series of courses provides hands-on experience with real computers and electronic devices. Courses and seminars are provided on a regular basis, and are also provided for groups, companies and schools at a site of their choosing. We are strong believers in practical laboratory exercises, so much time is spent working with electronic equipment, computers and circuits.

Additional information may be obtained from Dr. Chris Titus, the Blacksburg Group, Inc. (703) 951-9030 or from Dr. Linda Leffel, Virginia Tech Continuing Education Center (703) 961-5241.

Our group members are Mr. David G. Larsen, who is on the faculty of the Department of Chemistry at Virginia Tech, and Drs. Jon Titus and Chris Titus who work full-time with The Blacksburg Group, all of Blacksburg, VA.

TRS-80®
MORE THAN BASIC

This book presents a monitor program that makes a TRS-80 Model I or III microcomputer into a development system.

- The TRS-80 can be converted by loading object code from cassette or diskette, or by ROM replacement.
- The monitor executes valid instructions or commands, and it flags errors.
- The development system can be used to program in Z-80 mnemonics.
- Over 26 commands are available; the user is given total documentation.

The book also

- Describes hardware for a single-stepping feature.
- Discusses hardware for programming most of the popular EPROMs.
- Provides source codes for the monitor and the EPROM programmer.
- Presents hardware and software features of the TRS-80 Models I and III.

Howard W. Sams & Co., Inc.
4300 W. 62nd Street, Indianapolis, Indiana 46268 USA

ISBN: 0-672-21813-5